OMRON

Machine Automation Controller

NX-series

Safety Control Unit

Instructions Reference Manual

NX-SL□□□□

SYSMAC always in control

Z931-E1-05

NOTE

No patent liability is assumed with respect to the use of the information contained herein. Moreover, because OMRON is constantly striving to improve its high-quality products, the information contained in this manual is subject to change without notice. Every precaution has been taken in the preparation of this manual. Nevertheless, OMRON assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

Trademarks

- Sysmac and SYSMAC are trademarks or registered trademarks of OMRON Corporation in Japan and other countries for OMRON factory automation products.
- Microsoft, Windows, Excel, and Visual Basic are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.
- EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
- Safety over EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
- ODVA, CIP, CompoNet, DeviceNet, and EtherNet/IP are trademarks of ODVA.
- The SD and SDHC logos are trademarks of SD-3C, LLC.

Other company names and product names in this document are the trademarks or registered trademarks of their respective companies.

Copyrights

Microsoft product screen shots reprinted with permission from Microsoft Corporation.

Introduction

Thank you for purchasing Machine Automation Controller NX-series Safety Control Units.

This manual contains information that is necessary to use the NX-series Safety Control Units. Please read this manual and make sure you understand the functionality and performance of the NX-series Safety Control Units before you attempt to use them in a control system.

Keep this manual in a safe place where it will be available for reference during operation.

Intended Audience

This manual is intended for the following personnel, who must also have knowledge of electrical systems (an electrical engineer or the equivalent).

- Personnel in charge of introducing FA systems.
- Personnel in charge of designing FA systems.
- Personnel in charge of installing and maintaining FA systems.
- Personnel in charge of managing FA systems and facilities.
- Personnel with the qualifications, authority, and responsibility for providing safety at each phase of the lifecycle of the machine: design, installation, operation, maintenance, and disposal.
- Personnel with a knowledge of functional safety.

For programming, this manual is intended for personnel who understand the programming language specifications in international standard IEC 61131-3 or Japanese standard JIS B 3503.

Applicable Products

This manual covers the following products.

- NX-series Safety Control Units

  NX-SL□□□□

CONTENTS

Section 2 Standard Functions

Section 3 Safety Standard Function Blocks

Section 4 Safety Function Blocks

Index 2

Relevant Manuals

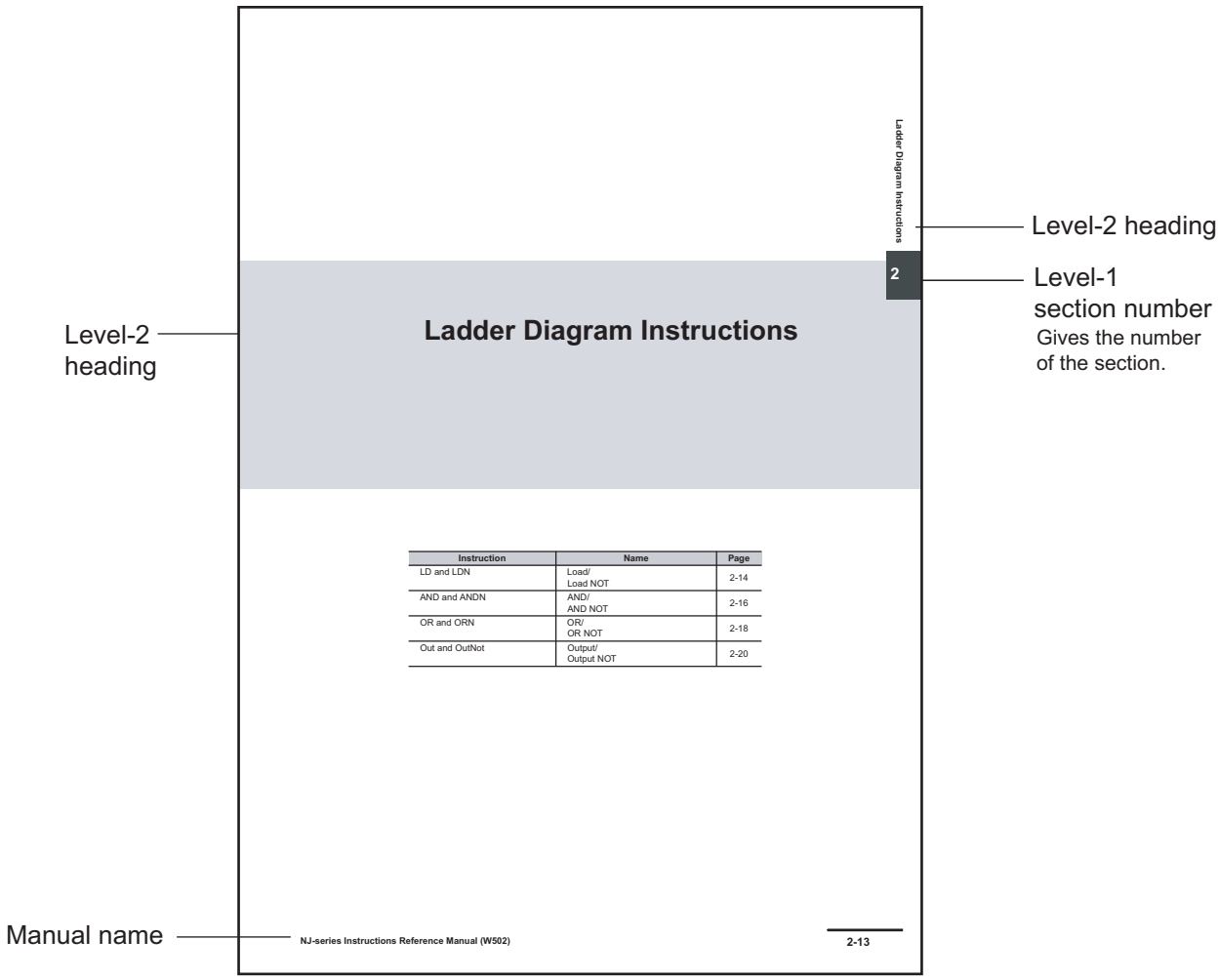
The information for this product is divided between two manuals as shown in the following table.
Read all of the manuals that are relevant to your system configuration and application before you use the product. Most operations are performed from the Sysmac Studio Automation Software.
Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for information on the Sysmac Studio.

Purpose of use	NX-series Safety Control Unit User's Manual	NX-series Safety Control Unit Instructions Reference Manual
Learning about Safety Control Units	●	
Mounting, installing, and making hardware settings for Safety Control Units	●	
Making software settings for Safety Control Units	●	
Creating safety programs	●	●
Verifying and debugging safety programs	●	●
Troubleshooting Safety Control Units	●	
Maintaining Safety Control Units	●	

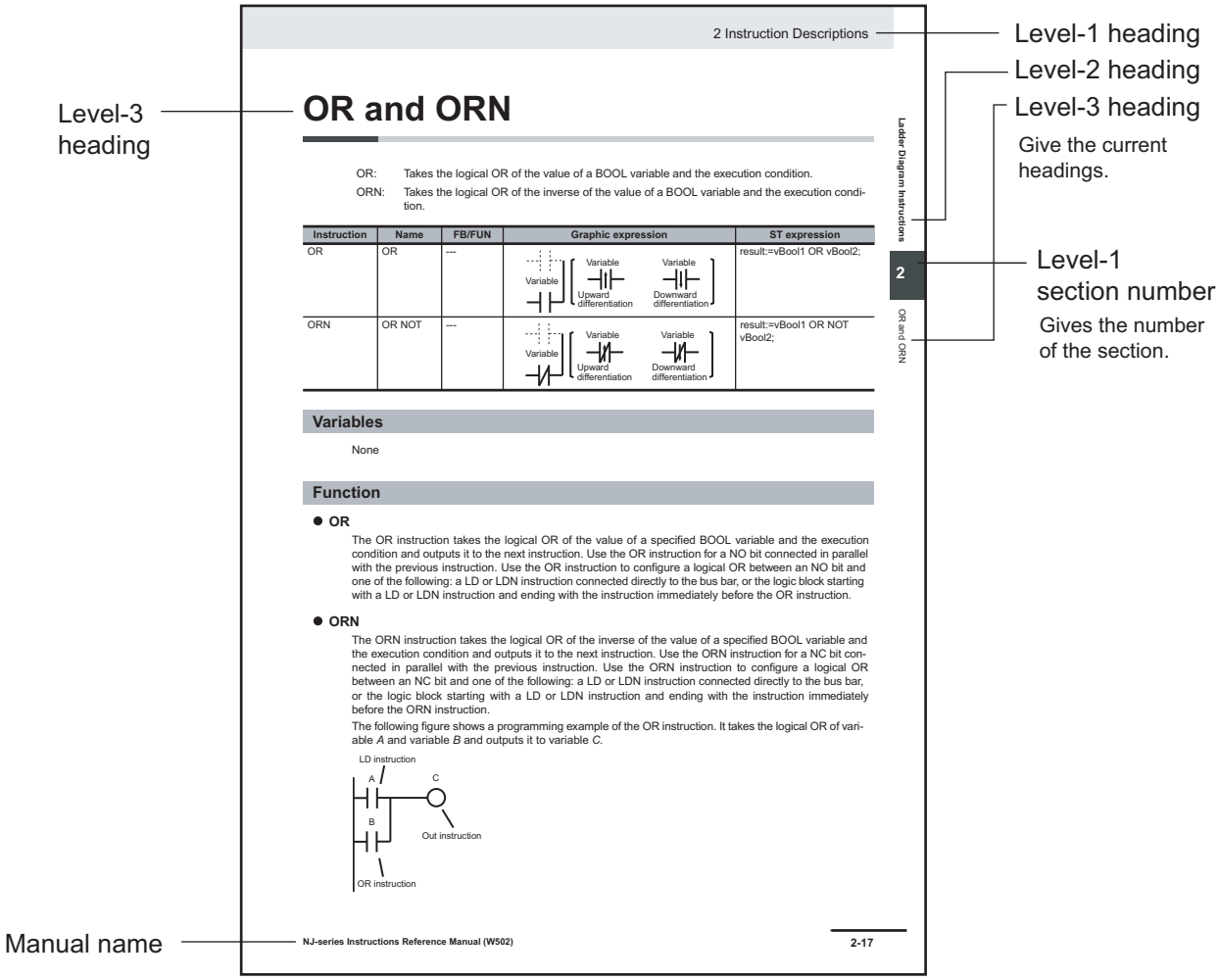
Manual Structure

Page Structure

The following page structure is used in this manual.



This page is for illustration only. It may not literally appear in this manual.



This page is for illustration only. It may not literally appear in this manual.

Special Information

Special information in this manual is classified as follows:

Additional Information

References are provided to more detailed or related information.

Version Information

Information on differences in specifications and functionality for CPU Units and EtherCAT Coupler Units with different unit versions and for different versions of the Sysmac Studio is given.

Terms and Conditions Agreement

Warranty, Limitations of Liability

Warranties

- **Exclusive Warranty**

  Omron's exclusive warranty is that the Products will be free from defects in materials and workmanship for a period of twelve months from the date of sale by Omron (or such other period expressed in writing by Omron). Omron disclaims all other warranties, express or implied.

- **Limitations**

  OMRON MAKES NO WARRANTY OR REPRESENTATION, EXPRESS OR IMPLIED, ABOUT NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OF THE PRODUCTS. BUYER ACKNOWLEDGES THAT IT ALONE HAS DETERMINED THAT THE PRODUCTS WILL SUITABLY MEET THE REQUIREMENTS OF THEIR INTENDED USE.

  Omron further disclaims all warranties and responsibility of any type for claims or expenses based on infringement by the Products or otherwise of any intellectual property right.

- **Buyer Remedy**

  Omron's sole obligation hereunder shall be, at Omron's election, to (i) replace (in the form originally shipped with Buyer responsible for labor charges for removal or replacement thereof) the non-complying Product, (ii) repair the non-complying Product, or (iii) repay or credit Buyer an amount equal to the purchase price of the non-complying Product; provided that in no event shall Omron be responsible for warranty, repair, indemnity or any other claims or expenses regarding the Products unless Omron's analysis confirms that the Products were properly handled, stored, installed and maintained and not subject to contamination, abuse, misuse or inappropriate modification. Return of any Products by Buyer must be approved in writing by Omron before shipment. Omron Companies shall not be liable for the suitability or unsuitability or the results from the use of Products in combination with any electrical or electronic components, circuits, system assemblies or any other materials or substances or environments. Any advice, recommendations or information given orally or in writing, are not to be construed as an amendment or addition to the above warranty.

See http://www.omron.com/global/ or contact your Omron representative for published information.

Limitation on Liability; Etc

OMRON COMPANIES SHALL NOT BE LIABLE FOR SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS OR PRODUCTION OR COMMERCIAL LOSS IN ANY WAY CONNECTED WITH THE PRODUCTS, WHETHER SUCH CLAIM IS BASED IN CONTRACT, WARRANTY, NEGLIGENCE OR STRICT LIABILITY.

Further, in no event shall liability of Omron Companies exceed the individual price of the Product on which liability is asserted.

Application Considerations

Suitability of Use

Omron Companies shall not be responsible for conformity with any standards, codes or regulations which apply to the combination of the Product in the Buyer's application or use of the Product. At Buyer's request, Omron will provide applicable third party certification documents identifying ratings and limitations of use which apply to the Product. This information by itself is not sufficient for a complete determination of the suitability of the Product in combination with the end product, machine, system, or other application or use. Buyer shall be solely responsible for determining appropriateness of the particular Product with respect to Buyer's application, product or system. Buyer shall take application responsibility in all cases.

NEVER USE THE PRODUCT FOR AN APPLICATION INVOLVING SERIOUS RISK TO LIFE OR PROPERTY OR IN LARGE QUANTITIES WITHOUT ENSURING THAT THE SYSTEM AS A WHOLE HAS BEEN DESIGNED TO ADDRESS THE RISKS, AND THAT THE OMRON PRODUCT(S) IS PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.

Programmable Products

Omron Companies shall not be responsible for the user's programming of a programmable Product, or any consequence thereof.

Disclaimers

Performance Data

Data presented in Omron Company websites, catalogs and other materials is provided as a guide for the user in determining suitability and does not constitute a warranty. It may represent the result of Omron's test conditions, and the user must correlate it to actual application requirements. Actual performance is subject to the Omron's Warranty and Limitations of Liability.

Change in Specifications

Product specifications and accessories may be changed at any time based on improvements and other reasons. It is our practice to change part numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the Product may be changed without any notice. When in doubt, special part numbers may be assigned to fix or establish key specifications for your application. Please consult with your Omron's representative at any time to confirm actual specifications of purchased Product.

Errors and Omissions

Information presented by Omron Companies has been checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical or proofreading errors or omissions.

Safety Precautions

Refer to the following manual for safety precautions.

- NX-series Safety Control Unit User's Manual (Cat No. Z930)

Precautions for Safe Use

Refer to the following manual for precautions for the safe use of the Safety Control Unit.

- NX-series Safety Control Unit User's Manual (Cat No. Z930)

Precautions for Correct Use

Refer to the following manual for precautions for the correct use of the Safety Control Unit.

- NX-series Safety Control Unit User's Manual (Cat No. Z930)

Regulations and Standards

The NX-series Safety Control Units are certified for the following standards.

Certification body	Standards	
TÜV Rheinland*1	• EN ISO 13849-1:2008 + AC:2009	• EN 61000-6-2:2005
	• EN ISO 13849-2:2012	• EN 61000-6-4:2007
	• IEC 61508 parts 1-7:2010	• NFPA 79:2012
	• EN 62061:2005	• ANSI RIA 15.06-1999
	• EN 61131-2:2007	• ANSI B11.19-2010
	• EN ISO 13850:2008	• UL1998
	• EN 60204-1:2006 + A1:2009 + AC:2010	• IEC 61326-3-1:2008
UL	• cULus: Listed (UL508) and ANSI/ISA 12.12.01	

*1. Certification was received for applications in which OMRON FSoE devices are connected to each other.

The NX-series Safety Control Units allow you to build a safety control system that meets the following standards.

- Requirements for SIL 3 (Safety Integrity Level 3) in IEC 61508, EN 62061 (Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems)
- Requirements for PLe (Performance Level e) and for safety category 4 in EN ISO 13849-1

The NX-series Safety Control Units are also registered for C-Tick and KC compliance.

Conformance to EC Directives

Applicable Directives

- EMC Directive
- Machinery Directive

Concepts

EMC Directives

OMRON devices that comply with EC Directives also conform to the related EMC standards so that they can be more easily built into other devices or the overall machine. The actual products have been checked for conformity to EMC standards.*1

Whether the products conform to the standards in the system used by the customer, however, must be checked by the customer. EMC-related performance of the OMRON devices that comply with EC Directives will vary depending on the configuration, wiring, and other conditions of the equipment or control panel on which the OMRON devices are installed. The customer must, therefore, perform the final check to confirm that devices and the overall machine conform to EMC standards.

*1. Applicable EMC (Electromagnetic Compatibility) standards are as follows:
EMS (Electromagnetic Susceptibility): EN 61131-2
EMI (Electromagnetic Interference): EN 61131-2 (Radiated emission: 10-m regulations).

Machinery Directive

The Machinery Directive demands that the safety components that are used to provide safety for the relevant machinery are used according to the required levels of safety.

The applicable directives are EN ISO 13849-1:2008 and EN 62061 SIL CL3.

Conformance to EC Directives

The NX-series Units comply with EC Directives. To ensure that the machine or device in which the NX-series Units are used complies with EC Directives, the following precautions must be observed.

- The NX-series Units must be installed within a metallic control cabinet.
- You must meet the following conditions for the DC power supplies that are connected as the Unit power supplies and I/O power supplies for the NX-series Units.

  (a) Use reinforced insulation or double insulation.

  (b) Ensure an output hold time of 20 ms min.

  (c) Use an SELV power supply that meets the requirements of IEC/EN 60950-1 and EN 50178.
  Do not allow the power supply cable length to exceed 3 m.

  We recommend that you use the OMRON S8JX-series Power Supplies. EMC standard compliance was confirmed for the recommended Power Supplies.
- NX-series Units that comply with EC Directives also conform to the Common Emission Standard (EN 61131-2). Radiated emission characteristics (10-m regulations) may vary depending on the configuration of the control panel used, other devices connected to the control panel, wiring, and other conditions.

  You must therefore confirm that the overall machine or equipment in which the NX-series Units are used complies with EC Directives.
- This is a Class A product (for industrial environments). In a residential environment, it may cause radio interference. If radio interference occurs, the user may be required to take appropriate measures.

Conformance to EN ISO 13849-1 and EN 62061

International standards EN ISO 13849-1 and EN 62061 demand that process controls be in place for the creation of safety-related software when building a safety control system that uses Safety Control Units. The process control must ensure that the software is easy to read, understand, test, and maintain to avoid system failures during each phase (i.e., general software design, safety circuit system design, and software upgrades) of the software design lifecycle.

This means that process controls must also be in place for the design and development of safety software, such as for equipment and machinery that use function blocks that are provided by the Safety Control Units.

It is the customer’s responsibility to conform with all standards.

Conformance to UL and CSA Standards

The NX-series Safety Control Units comply with the following UL and CSA standards. The application conditions for standard compliance are defined. Refer to the *Instruction Sheet* that is provided with each Unit before application.

Conformance to KC Certification

Refer to the following manual for conformance to KC certification.

- NX-series Safety Control Unit User's Manual (Cat No. Z930)

Software Licenses and Copyrights

This product incorporates certain third party software. The license and copyright information associated with this software is available at http://www.fa.omron.co.jp/nj_info_e/.

Unit Versions

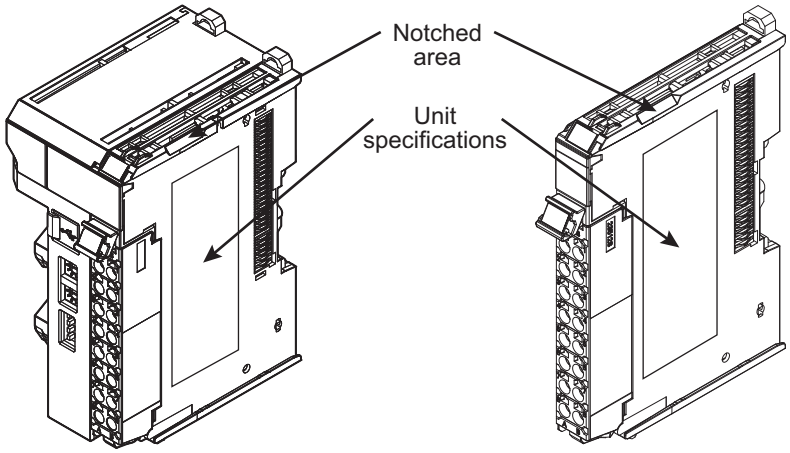
This section describes the notation that is used for unit versions, the confirmation method for unit versions, and the relationship between unit versions and Sysmac Studio versions.

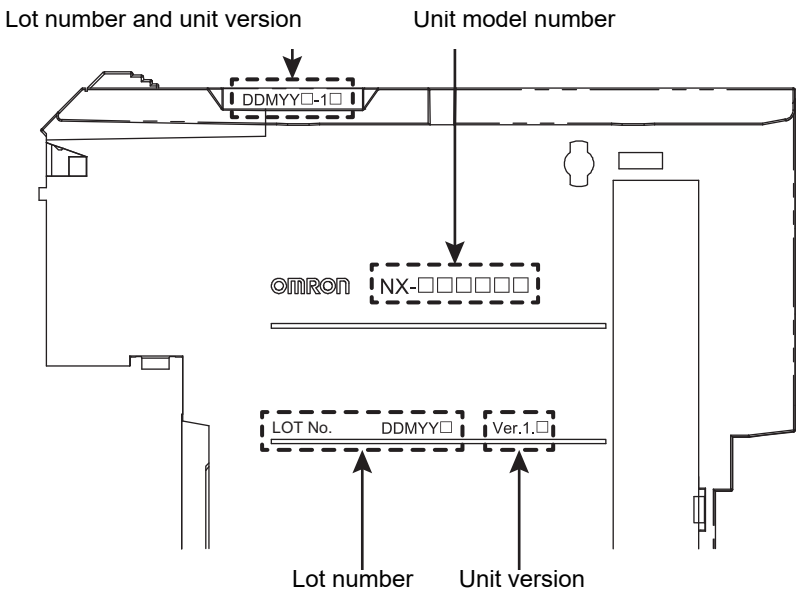
Unit Versions

A “unit version” has been introduced to manage the Units in the NX Series according to differences in functionality accompanying Unit upgrades.

Notation of Unit Versions on Products

The unit version is given with the Unit specifications on the side of the Unit or in the notched area.





The following information is provided in the Unit specifications on the Unit.

Name	Function
Unit model number	Gives the model of the Unit.
Unit version	Gives the unit version of the Unit.
Lot number	Gives the lot number of the Unit. DDMYY□: Lot number, □: Used by OMRON. “M” gives the month (1 to 9: January to September, X: October, Y: November, Z: December)

The following information is provided in the notched area on the Unit.

Name	Function
Lot number and unit version	Gives the lot number and unit version of the Unit. • DDMYY☐: Lot number, ☐: Used by OMRON. “M” gives the month (1 to 9: January to September, X: October, Y: November, Z: December) • 1☐: Unit version The decimal portion of the unit version is omitted. (It is provided in the Unit specifications.)

Confirming Unit Versions with the Sysmac Studio

You can use the Unit Production Information on the Sysmac Studio to check the unit versions EtherCAT Coupler Unit and NX Units.

1 Double-click **EtherCAT** under **Configurations and Setup** in the Multiview Explorer, and then double-click the EtherCAT Coupler Unit. Or, right-click the EtherCAT Coupler Unit and select ***Edit*** from the menu.
The Edit Slave Terminal Configuration Tab Page is displayed.

You can also display the Edit Slave Terminal Configuration Tab Page with any of the following operations.

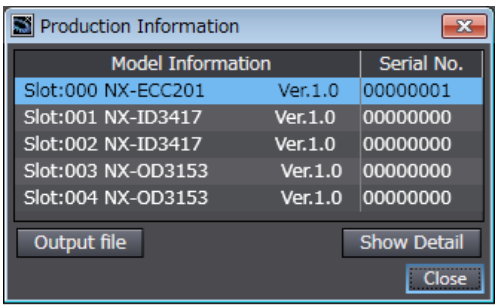
Double-click **EtherCAT** under **Configurations and Setup** in the Multiview Explorer, right-click the EtherCAT Coupler Unit in the EtherCAT Configuration Edit Tab Page, and select ***Edit Slave Terminal Configuration***.

Or, select the EtherCAT Coupler Unit on the EtherCAT Configuration Edit Tab Page click the **Edit Slave Terminal Configuration** Button.

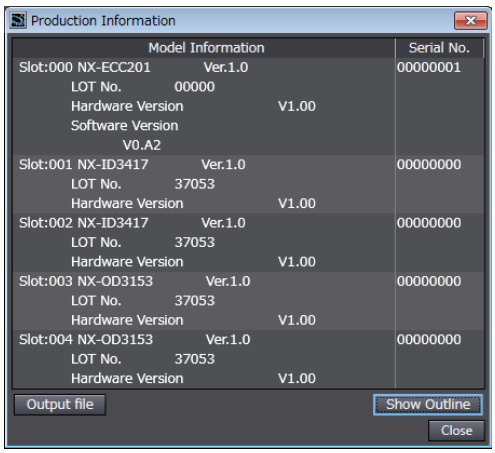
2 Go online.

3 Right-click the EtherCAT Coupler Unit and select ***Display Production Information*** from the menu.

The Production Information Dialog Box is displayed.



Simple Display



Detailed Display

In this example, “Ver.1.0” is displayed next to the Unit model.

The following items are displayed.

- Slot number
- Unit model number
- Unit version
- Serial number
- Lot number

- Hardware version
- Software version

The software version is displayed only for Units that contain software.

Unit Versions and Sysmac Studio Versions

The functions that are supported depend on the unit version of the Unit. The version of Sysmac Studio that supports the functions that were added for an upgrade is also required to use those functions.

Refer to the *NX-series Safety Control Unit User's Manual* (Cat. No. Z930) for the relationship between the unit versions of the CPU Units and the Sysmac Studio versions, and for the functions that are supported by each unit version.

Unit Version Notation

In this User's Manual, unit versions are specified as shown in the following table.

Unit version in Unit specifications on the product	Notation in this manual	Remarks
Ver. 1.□ or later	Unit version 1.0 or later	Unless unit versions are specified, the information in this manual applies to all unit versions.

Related Manuals

The following manuals are related. Use these manuals for reference.

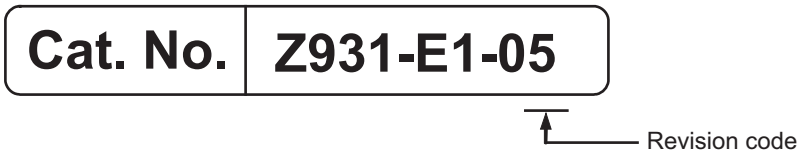
Manual name	Cat. No.	Model numbers	Application	Description
NX-series Safety Control Unit Instructions Reference Manual	Z931	NX-SL□□□□	Learning about the specifications of instructions for the Safety CPU Unit.	The instructions for the Safety CPU Unit are described. When programming, use this manual together with the *NX-series Safety Control Unit User's Manual* (Cat. No. Z930).
NX-series Safety Control Unit User's Manual	Z930	NX-SL□□□□ NX-SI□□□□ NX-SO□□□□	Learning how to use NX-series Safety Control Units.	The hardware, setup methods, and functions of the NX-series Safety Control Unit are described.
Sysmac Studio Version 1 Operation Manual	W504	SYSMAC-SE2□□□	Learning about the operating procedures and functions of the Sysmac Studio.	Describes the operating procedures of the Sysmac Studio.

Terminology

Refer to the *NX-series Safety Control Unit User's Manual* (Cat. No. Z930) for the definitions of terms that are used in this manual.

Revision History

A manual revision code appears as a suffix to the catalog number at the bottom left of the front and back covers of the manual.



Revision code	Date	Revised content
01	June 2013	Original production
02	September 2013	Corrected mistakes.
03	December 2013	Added information on timer set values. Corrected mistakes.
04	April 2016	• Changed Section 2 Other Standard Instructions to Selection Instructions. • Added *DigitalCode (decimal)* to *FB-specific Error Codes* and *FB-specific State Codes* tables in *Section 4 Safety Function Blocks*.
05	July 2019	Corrected mistakes.

Sections in this Manual

1

Introduction to Instructions for Safety Control Units and Interpreting Instruction

This section provides an introduction to the instructions for Safety Control Units and tells how to interpret the instruction descriptions.

1-1 Types of Instructions

The following three types of instructions can be used with the Safety Control Units.

Type	Description
Standard functions	These function instructions do not use safety data. They include program execution control instructions, data type conversion instructions, Boolean operation instructions, math instructions, comparison instructions, etc.
Safety standard function blocks	These function block instructions use safety data. They include counter instructions, up/down trigger instructions, timer instructions, etc.
Safety function blocks	These function block instructions use safety data and are based on the safety standards defined by PLCopen. They include an antivalent instruction, emergency stop instruction, etc.

Instruction specifications are provided starting from section 2.

Precautions for Correct Use

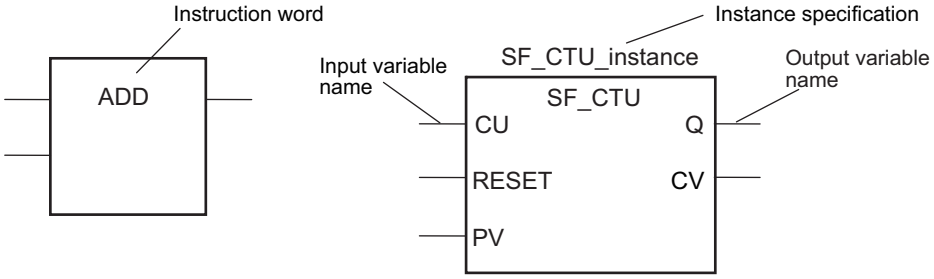
Some of the instructions have the same names as the instructions that are supported by the NJ-series CPU Units. Operating specifications, however, are different.

1-2 Interpreting Instruction Descriptions

The notation that is used to describe instructions is explained in this section.

1-2-1 Items

The following items are provided. The order of the items is not the same for all instructions. If there are items that are specific to one type of instruction, they are explained in the section for each instruction type.

Item	Description
Instruction	The instruction word is given.
Name	The name of the instruction is given.
FB/FUN	Whether the instruction is a function block (FB) instruction or a function (FUN) instruction is given.
Graphic expression	The figure that represents the instruction in a function block diagram is given. ● **Example for a FUN Instruction** ● **Example for a FB Instruction**  Instance specification: An instance of an instruction is indicated by “XX_instance” above a FB instruction. You must assign an instance name to any instance of an instruction that you specify.
Variables	• Variable The input variable or output variable is given. • Name The name of the variable is given. Example: Up-counter • I/O Whether the variable is an input variable or output variable is given. • Description The meaning of the variable and any restrictions are given. • Valid range The range that the variable can take is given. “Depends on data type” indicates that the valid range of the variable depends on the data type that you use. The valid ranges of the data types are given later in this section. • Default The specified default value is automatically used for the variable if you do not assign a parameter to the instruction before it is executed. “---” indicates the following: Input variables: The default value of the data type of the input variable is assigned. The default values of the data types are given later in this section. Output variables: Default values are not set. • Data type The data type of the variable is given. Broadly speaking, there are two classifications of data types: safety signals and non-safety signals. These two classifications of data types are described later.

Item	Description
Function	The function of the instruction is described.
Additional Information	Additional information on the function of the instruction is provided. This includes related instructions and helpful information for application of the instruction.
Precautions for Correct Use	Precautions for application of the instruction are given. The conditions under which errors occur for the instruction are also given here.

1-2-2 Safety Data Types and Standard Data Types

The Safety Control Unit classifies the following two data types to distinguish between safety signals and standard signals.

- Safety data types: These data types represent signals related to safety control.
- Standard data types: These data types represent signals related to standard control.

The safety data type variables are prefixed with the “SAFE” before the name of the standard data type, as in SAFEBOOL and SAFEBYTE.

You can input a signal for a safety data type variable to a standard data type variable. You cannot input a signal for a standard data type variable to a safety data type variable. A building error will occur.

1-2-3 Valid Ranges and Default Values of Variables

The valid range of a variable indicates the range of values that variable can take. The default value of a variable indicates the value that is assigned to an input variable when the instruction is executed without a parameter assigned to the input variable. These values are defined for each data type. If specific values are not given for an instruction, then the valid ranges and default values of the data types are applied. These variables are indicated by “depends on data type” in the valid range column of the table that describes the variables and by “---” in the input variable default column. The valid ranges and default values of the data types are given in the following tables.

Classification	Safety/standard data type	Data type	Range of values	Default
Boolean	**Standard data type**	**BOOL**	TRUE or FALSE	FALSE
	Safety data type	**SAFEBOOL**		
Bit strings	**Standard data type**	**BYTE**	byte#16#00 to byte#16#FF	byte#16#00
	Safety data type	**SAFEBYTE**		
	Standard data type	**WORD**	word#16#0000 to word#16#FFFF	word#16#0000
	Safety data type	**SAFEWORD**		
	Standard data type	**DWORD**	dword#16#00000000 to dword#16#FFFFFFFF	dword#16#00000000
	Safety data type	**SAFEDWORD**		
Integers	**Standard data type**	**INT**	int#−32768 to int#32767	int#0
	Safety data type	**SAFEINT**		
	Standard data type	**DINT**	dint#-2147483648 to dint#2147483647	dint#0
	Safety data type	**SAFEDINT**		
Durations	**Standard data type**	**TIME**	t#0ms to t#2147483647ms and t#0d0h0m0s0ms to t#24d20h31m23s647ms	t#0s
	Safety data type	**SAFETIME**		

1-2-4 Timer Set Values

Time set values, such as those for *DiscrepancyTime* or the OFF-Delay Timer instruction, operate in increments of the safety task period.

The timer error is +1 safety task period. Every safety task period, a timer value is checked to see if it has reached the set time. If the timer value reaches the set time immediately after this check, the time is delayed by one safety task period.

Examples are provided below.

● When the OFF-Delay Timer Instruction Is Set to 500 ms and the Safety Task Period Is Set to 16 ms

The timer will time out 512 ms (16 ms × 32) after the safety task is started.

● When the OFF-Delay Timer Instruction Is Set to 500 ms and the Safety Task Period Is Set to 20 ms

The timer will time out 520 ms (20 ms × 26) after the safety task is started. Care is required because the timer will not operate at 500 ms.

Refer to the *NX-series Safety Control Unit User's Manual* (Cat. No. Z930) for details on the safety task period.

Precautions for Correct Use

The time when an OFF-Delay Timer instruction times out can affect the safety reaction times.

2

Standard Functions

This section gives the specifications of the standard functions that you can use for NX-series Safety Control Units.

Table of Standard Functions

Type		Instruction	Name	Description	Page
Execution control		Jump	Jump	Moves processing to the jump destination specified by a label.	P. 2-6
		Return	Return	Returns control to the process that called the POU without executing any processing after RETURN.	P. 2-8
Data type conversion	Boolean to integer	BOOL_TO_INT	Convert BOOL to INT	Converts a BOOL variable to an INT variable.	P. 2-12
		BOOL_TO_DINT	Convert BOOL to DINT	Converts a BOOL variable to a DINT variable.	P. 2-13
	Boolean to duration	BOOL_TO_TIME	Convert BOOL to TIME	Converts a BOOL variable to a TIME variable.	P. 2-14
	Boolean to bit string	BOOL_TO_WORD	Convert BOOL to WORD	Converts a BOOL variable to a WORD variable.	P. 2-15
	Bit string to integer	BYTE_TO_INT	Convert BYTE to INT	Converts a BYTE variable to an INT variable.	P. 2-16
		BYTE_TO_DINT	Convert BYTE to DINT	Converts a BYTE variable to a DINT variable.	P. 2-17
		WORD_TO_INT	Convert WORD to INT	Converts a WORD variable to an INT variable.	P. 2-18
		WORD_TO_DINT	Convert WORD to DINT	Converts a WORD variable to a DINT variable.	P. 2-20
		DWORD_TO_DINT	Convert DWORD to DINT	Converts a DWORD variable to a DINT variable.	P. 2-21
	Bit string to duration	BYTE_TO_TIME	Convert BYTE to TIME	Converts a BYTE variable to a TIME variable.	P. 2-23
		WORD_TO_TIME	Convert WORD to TIME	Converts a WORD variable to a TIME variable.	P. 2-24
		DWORD_TO_TIME	Convert DWORD to TIME	Converts a DWORD variable to a TIME variable.	P. 2-25
	Bit string to bit string	BYTE_TO _WORD	Convert BYTE to WORD	Converts a BYTE variable to a WORD variable.	P. 2-26
		WORD_TO_BYTE	Convert WORD to BYTE	Converts a WORD variable to a BYTE variable.	P. 2-27
		WORD_TO_DWORD	Convert WORD to DWORD	Converts a WORD variable to a DWORD variable.	P. 2-28

Type		Instruction	Name	Description	Page
Data type conversion	Integer to Boolean	DINT_TO_BOOL	Convert DINT to BOOL	Converts a DINT variable to a BOOL variable.	P. 2-29
		INT_TO_BOOL	Convert INT to BOOL	Converts an INT variable to a BOOL variable.	P. 2-30
	Integer to bit string	DINT_TO_BYTE	Convert DINT to BYTE	Converts a DINT variable to a BYTE variable.	P. 2-31
		DINT_TO_DWORD	Convert DINT to DWORD	Converts a DINT variable to a DWORD variable.	P. 2-32
		DINT_TO_WORD	Convert DINT to WORD	Converts a DINT variable to a WORD variable.	P. 2-34
		INT_TO_BYTE	Convert INT to BYTE	Converts an INT variable to a BYTE variable.	P. 2-35
		INT_TO_DWORD	Convert INT to DWORD	Converts an INT variable to a DWORD variable.	P. 2-36
		INT_TO_WORD	Convert INT to WORD	Converts an INT variable to a WORD variable.	P. 2-38
	Integer to integer	DINT_TO_INT	Convert DINT to INT	Converts a DINT variable to an INT variable.	P. 2-40
		INT_TO_DINT	Convert INT to DINT	Converts an INT variable to a DINT variable.	P. 2-41
	Integer to duration	DINT_TO_TIME	Convert DINT to TIME	Converts a DINT variable to a TIME variable.	P. 2-42
		INT_TO_TIME	Convert INT to TIME	Converts an INT variable to a TIME variable.	P. 2-43
	Duration to Boolean	TIME_TO_BOOL	Convert TIME to BOOL	Converts a TIME variable to a BOOL variable.	P. 2-44
	Duration to bit string	TIME_TO_BYTE	Convert TIME to BYTE	Converts a TIME variable to a BYTE variable.	P. 2-45
		TIME_TO_DWORD	Convert TIME to DWORD	Converts a TIME variable to a DWORD variable.	P. 2-46
		TIME_TO_WORD	Convert TIME to WORD	Converts a TIME variable to a WORD variable.	P. 2-47
	Duration to integer	TIME_TO_DINT	Convert TIME to DINT	Converts a TIME variable to a DINT variable.	P. 2-48
		TIME_TO_INT	Convert TIME to INT	Converts a TIME variable to an INT variable.	P. 2-49
	Bit string to Boolean	WORD_TO_BOOL	Convert WORD to BOOL	Converts a WORD variable to a BOOL variable.	P. 2-50

Type	Instruction	Name	Description	Page
Boolean operations	AND	Logical AND	Performs a logical AND on multiple Boolean variables.	P. 2-52
	OR	Logical OR	Performs a logical OR on multiple Boolean variables.	P. 2-52
	XOR	Exclusive logical OR	Performs an exclusive logical OR on multiple Boolean variables.	P. 2-52
	NOT	Bit Reversal	Reverses the value of a Boolean bit.	P. 2-54
Math	ADD	Addition	Adds integers or durations.	P. 2-56
	SUB	Subtraction	Subtracts integers or durations.	P. 2-58
	MUL	Multiplication	Multiplies integers or a duration.	P. 2-60
	DIV	Division	Divides integers or a duration.	P. 2-62
Comparison	EQ	Equal	Determines if the values of two variables are equivalent.	P. 2-66
	NE	Not Equal	Determines if the values of two variables are not equivalent.	P. 2-67
	LT	Less Than	Performs a less than comparison between two values.	P. 2-68
	LE	Less Than Or Equal	Performs a less than or equal comparison between two values.	P. 2-68
	GT	Greater Than	Performs a greater than comparison between two values.	P. 2-68
	GE	Greater Than Or Equal	Performs a greater than or equal comparison between two values.	P. 2-68
Selection	SEL	Bit Selection	Selects one of two selections.	P. 2-72
	MUX	Multiplexer	Selects one of multiple selections.	P. 2-74

Execution Control Instructions

Type	Instruction	Name	Description	Page
Execution control	Jump	Jump	Moves processing to the jump destination specified by a label.	P. 2-6
	Return	Return	Returns control to the process that called the POU without executing any processing after RETURN.	P. 2-8

JUMP and LABEL

This function moves processing to the jump destination specified by a label.

Instruction	Name	FB/FUN	Graphic expression
JUMP	Jump	FUN	→Label
LABEL	Label	FUN	Label:

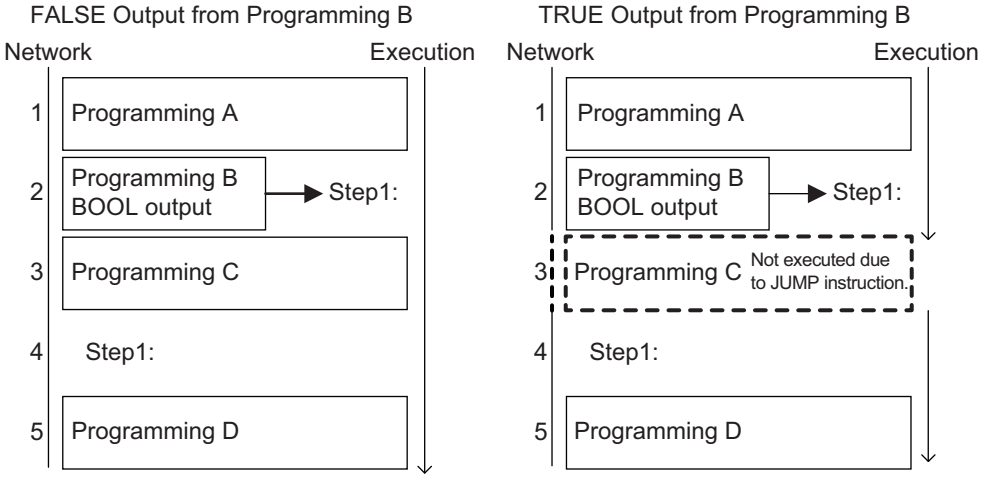
Variables

There are no variables for these instructions.

Function

When the execution condition is TRUE, the JUMP instruction moves processing to the jump destination specified by a label in the program.
The label can be any text string.

The following figure shows a programming example. This example uses the text string *Step1* as the label.
When the JUMP instruction is executed, processing moves to the location marked *Step1*. In this example, programming C between the JUMP instruction and the label is not executed. The outputs in programming C retain the values that they had just before the JUMP instruction was executed.



Additional Information

- You cannot jump upward in the networks.
- You can use the same label as the jump destination for more than one JUMP instruction.
- You can set only a label in a network, or you can set both programming and a label in a network.

Precautions for Correct Use

- You must use either a BOOL or SAFEBOOL execution condition for the JUMP instruction.
  If you connect an execution condition with any other data type, a building error will occur.
- You cannot omit labels. If you omit a label, a building error will occur.
- Place the JUMP instruction and label in the same POU.
- Programming between the JUMP instruction and the label is not executed when the JUMP instruction is executed. The outputs retain the values that they had just before the JUMP instruction was executed.

RETURN

This function returns control to the process that called the POU without executing any processing after RETURN.

Instruction	Name	FB/FUN	Graphic expression
RETURN	Return	FUN	—◀RETURN▶

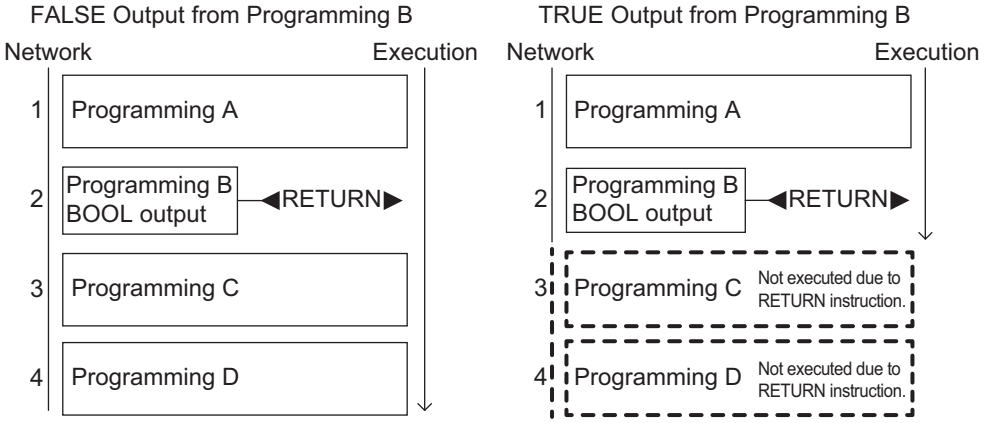
Variables

There are no variables for these instructions.

Function

When the execution condition is TRUE, control is returned to the location that called the POU without executing any processing after RETURN.

The following figure shows a programming example. When the RETURN instruction is executed in the example, programming C and D that follow it are not executed. The outputs in programming C and D retain the values that they had just before the RETURN instruction was executed.



Precautions for Correct Use

- You must use either a BOOL or SAFEBOOL execution condition for the RETURN instruction. If you connect an execution condition with any other data type, a building error will occur.
- If you use this instruction too often, the flow of processing will be difficult to understand. Use it with caution.
- Programming after the RETURN instruction is not executed when the RETURN instruction is executed. The outputs retain the values that they had just before the RETURN instruction was executed.

Data Type Conversion Instructions

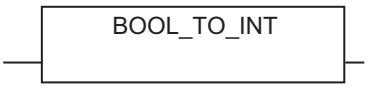
Type		Instruction	Name	Description	Page
Data type conversion	Boolean to integer	BOOL_TO_INT	Convert BOOL to INT	Converts a BOOL variable to an INT variable.	P. 2-12
		BOOL_TO_DINT	Convert BOOL to DINT	Converts a BOOL variable to a DINT variable.	P. 2-13
	Boolean to duration	BOOL_TO_TIME	Convert BOOL to TIME	Converts a BOOL variable to a TIME variable.	P. 2-14
	Boolean to bit string	BOOL_TO_WORD	Convert BOOL to WORD	Converts a BOOL variable to a WORD variable.	P. 2-15

Type		Instruction	Name	Description	Page
Data type conversion	Bit string to integer	BYTE_TO_INT	Convert BYTE to INT	Converts a BYTE variable to an INT variable.	P. 2-16
		BYTE_TO_DINT	Convert BYTE to DINT	Converts a BYTE variable to a DINT variable.	P. 2-17
		WORD_TO_INT	Convert WORD to INT	Converts a WORD variable to an INT variable.	P. 2-18
		WORD_TO_DINT	Convert WORD to DINT	Converts a WORD variable to a DINT variable.	P. 2-20
		DWORD_TO_DINT	Convert DWORD to DINT	Converts a DWORD variable to a DINT variable.	P. 2-21
	Bit string to duration	BYTE_TO_TIME	Convert BYTE to TIME	Converts a BYTE variable to a TIME variable.	P. 2-23
		WORD_TO_TIME	Convert WORD to TIME	Converts a WORD variable to a TIME variable.	P. 2-24
		DWORD_TO_TIME	Convert DWORD to TIME	Converts a DWORD variable to a TIME variable.	P. 2-25
	Bit string to bit string	BYTE_TO _WORD	Convert BYTE to WORD	Converts a BYTE variable to a WORD variable.	P. 2-26
		WORD_TO_BYTE	Convert WORD to BYTE	Converts a WORD variable to a BYTE variable.	P. 2-27
		WORD_TO_DWORD	Convert WORD to DWORD	Converts a WORD variable to a DWORD variable.	P. 2-28
	Integer to Boolean	DINT_TO_BOOL	Convert DINT to BOOL	Converts a DINT variable to a BOOL variable.	P. 2-29
		INT_TO_BOOL	Convert INT to BOOL	Converts an INT variable to a BOOL variable.	P. 2-30
	Integer to bit string	DINT_TO_BYTE	Convert DINT to BYTE	Converts a DINT variable to a BYTE variable.	P. 2-31
		DINT_TO_DWORD	Convert DINT to DWORD	Converts a DINT variable to a DWORD variable.	P. 2-32
		DINT_TO_WORD	Convert DINT to WORD	Converts a DINT variable to a WORD variable.	P. 2-34
		INT_TO_BYTE	Convert INT to BYTE	Converts an INT variable to a BYTE variable.	P. 2-35
		INT_TO_DWORD	Convert INT to DWORD	Converts an INT variable to a DWORD variable.	P. 2-36
		INT_TO_WORD	Convert INT to WORD	Converts an INT variable to a WORD variable.	P. 2-38
	Integer to integer	DINT_TO_INT	Convert DINT to INT	Converts a DINT variable to an INT variable.	P. 2-40
		INT_TO_DINT	Convert INT to DINT	Converts an INT variable to a DINT variable.	P. 2-41

Type		Instruction	Name	Description	Page
Data type conversion	Integer to duration	DINT_TO_TIME	Convert DINT to TIME	Converts a DINT variable to a TIME variable.	P. 2-42
		INT_TO_TIME	Convert INT to TIME	Converts an INT variable to a TIME variable.	P. 2-43
	Duration to Boolean	TIME_TO_BOOL	Convert TIME to BOOL	Converts a TIME variable to a BOOL variable.	P. 2-44
	Duration to bit string	TIME_TO_BYTE	Convert TIME to BYTE	Converts a TIME variable to a BYTE variable.	P. 2-45
		TIME_TO_DWORD	Convert TIME to DWORD	Converts a TIME variable to a DWORD variable.	P. 2-46
		TIME_TO_WORD	Convert TIME to WORD	Converts a TIME variable to a WORD variable.	P. 2-47
	Duration to integer	TIME_TO_DINT	Convert TIME to DINT	Converts a TIME variable to a DINT variable.	P. 2-48
		TIME_TO_INT	Convert TIME to INT	Converts a TIME variable to an INT variable.	P. 2-49
	Bit string to Boolean	WORD_TO_BOOL	Convert WORD to BOOL	Converts a WORD variable to a BOOL variable.	P. 2-50

BOOL_TO_INT

This function converts a BOOL variable to an INT variable.

Instruction	Name	FB/FUN	Graphic expression
BOOL_TO_INT	Convert BOOL to INT	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	TRUE or FALSE	FALSE
Out	Conversion result	Output	Conversion result	INT#0 or INT#1	INT#0

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	OK	OK												
Out									OK	OK				

Function

This function converts BOOL data *In* to INT data *Out*.

If the value of *In* is FALSE, the value of *Out* is INT#0.

If the value of *In* is TRUE, the value of *Out* is INT#1.

Additional Information

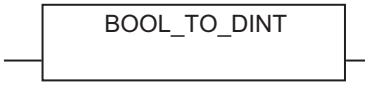
To check for INT data, refer to INT_TO_BOOL.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

BOOL_TO_DINT

This function converts a BOOL variable to a DINT variable.

Instruction	Name	FB/FUN	Graphic expression
BOOL_TO_DINT	Convert BOOL to DINT	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	TRUE or FALSE	FALSE
Out	Conversion result	Output	Conversion result	DINT#0 or DINT#1	DINT#0

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	OK	OK												
Out											OK	OK		

Function

This function converts BOOL data *In* to DINT data *Out*.

If the value of *In* is FALSE, the value of *Out* is DINT#0.

If the value of *In* is TRUE, the value of *Out* is DINT#1.

Additional Information

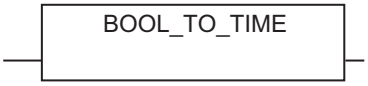
To check for DINT data, refer to DINT_TO_BOOL.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

BOOL_TO_TIME

This function converts a BOOL variable to a TIME variable.

Instruction	Name	FB/FUN	Graphic expression
BOOL_TO_TIME	Convert BOOL to TIME	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	TRUE or FALSE	FALSE
Out	Conversion result	Output	Duration	T#0ms or T#1ms	T#0ms

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	OK	OK												
Out													OK	OK

Function

This function converts BOOL data *In* to TIME data *Out*.

If the value of *In* is FALSE, the value of *Out* is 0 ms (T#0ms).

If the value of *In* is TRUE, the value of *Out* is 1 ms (T#1ms).

Additional Information

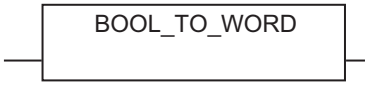
To check for TIME data, refer to TIME_TO_BOOL.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

BOOL_TO_WORD

This function converts a BOOL variable to a WORD variable.

Instruction	Name	FB/FUN	Graphic expression
BOOL_TO_WORD	Convert BOOL to WORD	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	TRUE or FALSE	FALSE
Out	Conversion result	Output	Conversion result	WORD#16#0000 or WORD#16#0001	WORD#16#0000

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	OK	OK												
Out					OK	OK								

Function

This function converts BOOL data *In* to WORD data *Out*.

If the value of *In* is FALSE, the value of *Out* is WORD#16#0000.

If the value of *In* is TRUE, the value of *Out* is WORD#16#0001.

Additional Information

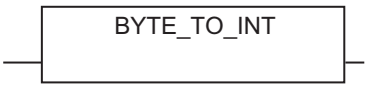
To check for WORD data, refer to WORD_TO_BOOL.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

BYTE_TO_INT

This function converts a BYTE variable to an INT variable.

Instruction	Name	FB/FUN	Graphic expression
BYTE_TO_INT	Convert BYTE to INT	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	BYTE#16#00 to FF	BYTE#16#00
Out	Conversion result	Output	Conversion result	INT#0 to 255	INT#0

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In			OK	OK										
Out									OK	OK				

Function

This function converts BYTE data *In* to INT data *Out*.

Additional Information

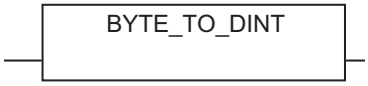
To convert INT data to BYTE data, refer to INT_TO_BYTE.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

BYTE_TO_DINT

This function converts a BYTE variable to a DINT variable.

Instruction	Name	FB/FUN	Graphic expression
BYTE_TO_DINT	Convert BYTE to DINT	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	BYTE#16#00 to FF	BYTE#16#00
Out	Conversion result	Output	Conversion result	DINT#0 to 255	DINT#0

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In			OK	OK										
Out											OK	OK		

Function

This function converts BYTE data *In* to DINT data *Out*.

Additional Information

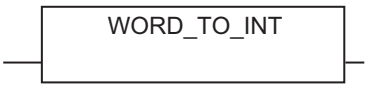
To convert DINT data to BYTE data, refer to DINT_TO_BYTE.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

WORD_TO_INT

This function converts a WORD variable to an INT variable.

Instruction	Name	FB/FUN	Graphic expression
WORD_TO_INT	Convert WORD to INT	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	WORD#16#0000 to FFFF	WORD#16#0000
Out	Conversion result	Output	Conversion result	INT#–32768 to 32767	INT#0

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In					OK	OK								
Out									OK	OK				

Function

This function converts WORD data *In* to INT data *Out*.

Example When Value of *Out* Is Positive (INT#0 to INT#32767)

- The value of *Out* is INT#0 to INT#32767 according to the value of *In* (WORD#16#0000 to WORD#16#7FFF).

Example When Value of *Out* Is Negative (INT#–32768 to INT#–1)

- If the value of *In* is WORD#16#8000 (1000 0000 0000 0000 binary), the value of *Out* is INT#32768, which is INT#–32768 as a 2-byte expression.
- If the value of *In* is WORD#16#FFFF (1111 1111 1111 1111 binary), the value of *Out* is INT#65535, which is INT#–1 as a 2-byte expression.

Additional Information

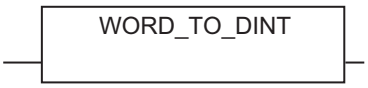
To convert INT data to WORD data, refer to INT_TO_WORD.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

WORD_TO_DINT

This function converts a WORD variable to a DINT variable.

Instruction	Name	FB/FUN	Graphic expression
WORD_TO_DINT	Convert WORD to DINT	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	WORD#16#0000 to FFFF	WORD#16#0000
Out	Conversion result	Output	Conversion result	DINT#0 to 65535	DINT#0

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In					OK	OK								
Out											OK	OK		

Function

This function converts WORD data *In* to DINT data *Out*.

Example for the Range of WORD Data (WORD#16#0000 to WORD#16#FFFF)

- The value of *Out* is DINT#0 to DINT#65535.

Additional Information

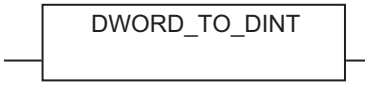
To convert DINT data to WORD data, refer to DINT_TO_WORD.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

DWORD_TO_DINT

This function converts a DWORD variable to a DINT variable.

Instruction	Name	FB/FUN	Graphic expression
DWORD_TO_DINT	Convert DWORD to DINT	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	DWORD#16#00000000 to FFFFFFFF	DWORD#16#00000000
Out	Conversion result	Output	Conversion result	DINT#–2147483648 to 2147483647	DINT#0

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In							OK	OK						
Out											OK	OK		

Function

This function converts DWORD data *In* to DINT data *Out*.

Example When Value of *Out* Is Positive (DINT#0 to DINT#2147483647)

- The value of *Out* is DINT#0 to DINT#2147483647 according to the value of *In* (DWORD#16#00000000 to DWORD#16#7FFFFFFF).

Example When Value of *Out* Is Negative (DINT#–2147483648 to DINT#–1)

- If the value of *In* is DWORD#16#80000000 (1000 0000 0000 0000 0000 0000 0000 0000 binary), the value of *Out* is DINT#–2147483648.
- If the value of *In* is DWORD#16#FFFFFFFF (1111 1111 1111 1111 1111 1111 1111 1111 binary), the value of *Out* is DINT#–1.

Additional Information

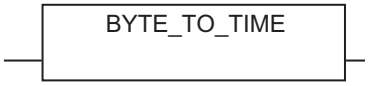
To convert DINT data to DWORD data, refer to DINT_TO_DWORD.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

BYTE_TO_TIME

This function converts a BYTE variable to a TIME variable.

Instruction	Name	FB/FUN	Graphic expression
BYTE_TO_TIME	Convert BYTE to TIME	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	BYTE#16#00 to FF	BYTE#16#00
Out	Conversion result	Output	Duration	T#0ms to T#255ms	T#0ms

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In			OK	OK										
Out													OK	OK

Function

This function converts BYTE data *In* to TIME data *Out.*

Additional Information

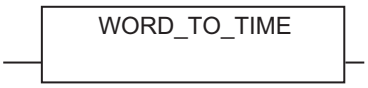
To convert TIME data to BYTE data, refer to TIME_TO_BYTE.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

WORD_TO_TIME

This function converts a WORD variable to a TIME variable.

Instruction	Name	FB/FUN	Graphic expression
WORD_TO_TIME	Convert WORD to TIME	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	WORD#16#0000 to FFFF	WORD#16#0000
Out	Conversion result	Output	Duration	T#0ms to T#1m5s535ms	T#0ms

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In					OK	OK								
Out													OK	OK

Function

This function converts WORD data *In* to TIME data *Out*.

Example When Value of *In* Is WORD#16#C549

- The value of *Out* is 50 s 505 ms (T#50s505ms).

Additional Information

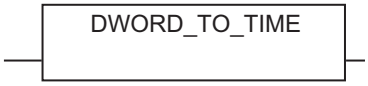
To convert TIME data to WORD data, refer to TIME_TO_WORD.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

DWORD_TO_TIME

This function converts a DWORD variable to a TIME variable.

Instruction	Name	FB/FUN	Graphic expression
DWORD_TO_TIME	Convert DWORD to TIME	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	DWORD#16#00000000 to FFFFFFFF	DWORD#16#00000000
Out	Conversion result	Output	Duration	T#0ms to T#49d17h2m47s295ms	T#0ms

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In							OK	OK						
Out													OK	OK

Function

This function converts DWORD data *In* to TIME data *Out.*

Example When Value of *In* Is DWORD#16#FFFFFFFF

- The value of *Out* is 49 days 17 h 2 min 47 s 295 ms (T#49d17h2m47s295ms).

Additional Information

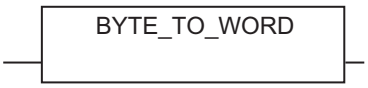
To convert TIME data to DWORD data, refer to TIME_TO_DWORD.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

BYTE_TO_WORD

This function converts a BYTE variable to a WORD variable.

Instruction	Name	FB/FUN	Graphic expression
BYTE_TO_WORD	Convert BYTE to WORD	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	BYTE#16#00 to FF	BYTE#16#00
Out	Conversion result	Output	Conversion result	WORD#16#0000 to 00FF	WORD#16#0000

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In			OK	OK										
Out					OK	OK								

Function

This function converts BYTE data *In* to WORD data *Out*.

Additional Information

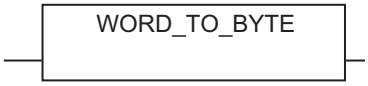
To convert WORD data to BYTE data, refer to WORD_TO_BYTE.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

WORD_TO_BYTE

This function converts a WORD variable to a BYTE variable.

Instruction	Name	FB/FUN	Graphic expression
WORD_TO_BYTE	Convert WORD to BYTE	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	WORD#16#0000 to 00FF	WORD#16#0000
Out	Conversion result	Output	Conversion result	BYTE#16#00 to FF	BYTE#16#00

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In					OK	OK								
Out			OK	OK										

Function

This function converts WORD data *In* to BYTE data *Out*.

Additional Information

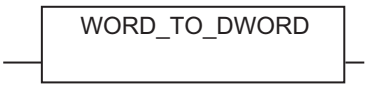
To convert BYTE data to WORD data, refer to BYTE_TO_WORD.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.
- If the input value is not between WORD#16#0000 and WORD#16#00FF, a cast error will occur and the program will stop.

WORD_TO_DWORD

This function converts a WORD variable to a DWORD variable.

Instruction	Name	FB/FUN	Graphic expression
WORD_TO_DWORD	Convert WORD to DWORD	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	WORD#16#0000 to FFFF	WORD#16#0000
Out	Conversion result	Output	Conversion result	DWORD#16#00000000 to 0000FFFF	DWORD#16#00000000

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In					OK	OK								
Out							OK	OK						

Function

This function converts WORD data *In* to DWORD data *Out*.

Example for the Range of WORD Data (WORD#16#0000 to WORD#16#FFFF)

- The value of *Out* will be DWORD#16#00000000 to DWORD#16#0000FFFF.

Additional Information

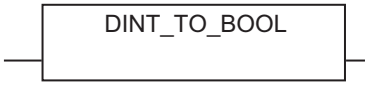
There is no instruction that converts DWORD data to WORD data.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

DINT_TO_BOOL

This function converts a DINT variable to a BOOL variable.

Instruction	Name	FB/FUN	Graphic expression
DINT_TO_BOOL	Convert DINT to BOOL	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	DINT#–2147483648 to 2147483647	DINT#0
Out	Conversion result	Output	Conversion result	TRUE or FALSE	FALSE

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In											OK	OK		
Out	OK	OK												

Function

This function converts DINT data *In* to BOOL data *Out*.

If the value of *In* is DINT#0, the value of *Out* is FALSE.

If the value of *In* is DINT#–2147483648 to DINT#–1 or DINT#1 to DINT#2147483647 (i.e., not DINT#0), the value of *Out* is TRUE.

Additional Information

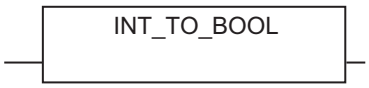
To convert BOOL data to DINT data, refer to BOOL_TO_DINT.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.
- If the input value is not DINT#-2147483648 to DINT#2147483647, a cast error will occur and the program will stop.

INT_TO_BOOL

This function converts an INT variable to a BOOL variable.

Instruction	Name	FB/FUN	Graphic expression
INT_TO_BOOL	Convert INT to BOOL	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	INT#–32768 to 32767	INT#0
Out	Conversion result	Output	Conversion result	TRUE or FALSE	FALSE

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In									OK	OK				
Out	OK	OK												

Function

This function converts INT data *In* to BOOL data *Out*.

If the value of *In* is INT#0, the value of *Out* is FALSE.

If the value of *In* is INT#–32768 to INT#–1 or INT#1 to INT#32767 (i.e., not INT#0), the value of *Out* is TRUE.

Additional Information

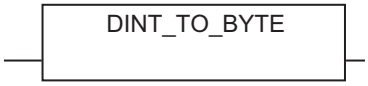
To convert BOOL data to INT data, refer to BOOL_TO_INT.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.
- If the input value is not INT#-32768 to INT#32767, a cast error will occur and the program will stop.

DINT_TO_BYTE

This function converts a DINT variable to a BYTE variable.

Instruction	Name	FB/FUN	Graphic expression
DINT_TO_BYTE	Convert DINT to BYTE	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	DINT#0 to 255	DINT#0
Out	Conversion result	Output	Conversion result	BYTE#16#00 to FF	BYTE#16#00

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In											OK	OK		
Out			OK	OK										

Function

This function converts DINT data *In* to BYTE data *Out*.

Additional Information

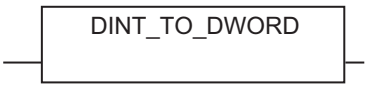
To convert BYTE data to DINT data, refer to BYTE_TO_DINT.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.
- If the input value is not DINT#0 to DINT#255, a cast error will occur and the program will stop.

DINT_TO_DWORD

This function converts a DINT variable to a DWORD variable.

Instruction	Name	FB/FUN	Graphic expression
DINT_TO_DWORD	Convert DINT to DWORD	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	DINT#–2147483648 to 2147483647	DINT#0
Out	Conversion result	Output	Conversion result	DWORD#16#00000000 to FFFFFFFF	DWORD#16#00000000

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In											OK	OK		
Out							OK	OK						

Function

This function converts DINT data *In* to DWORD data *Out*.

Example When Value of *In* Is Positive (DINT#0 to DINT#2147483647)

- The value of *Out* is DWORD#16#00000000 to DWORD#16#7FFFFFFF according to the value of *In* (DINT#0 to DINT#2147483647).

Example When Value of *In* Is Negative (DINT#–2147483648 to DINT#–1)

- If the value of *In* is DINT#–2147483648 (1000 0000 0000 0000 0000 0000 0000 0000 binary), the value of *Out* is DWORD#16#80000000.
- If the value of *In* is DINT#–1 (1111 1111 1111 1111 1111 1111 1111 1111 binary), the value of *Out* is DWORD#16#FFFFFFFF.

Additional Information

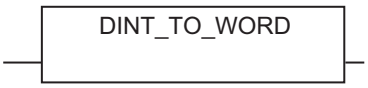
To convert DWORD data to DINT data, refer to DWORD_TO_DINT.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

DINT_TO_WORD

This function converts a DINT variable to a WORD variable.

Instruction	Name	FB/FUN	Graphic expression
DINT_TO_WORD	Convert DINT to WORD	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	DINT#0 to 65535	DINT#0
Out	Conversion result	Output	Conversion result	WORD#16#0000 to FFFF	WORD#16#0000

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In											OK	OK		
Out					OK	OK								

Function

This function converts DINT data *In* to WORD data *Out*.

Additional Information

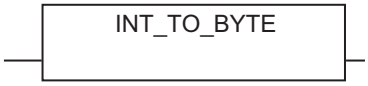
To convert WORD data to DINT data, refer to WORD_TO_DINT.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.
- If the input value is not DINT#0 to DINT#65535, a cast error will occur and the program will stop.

INT_TO_BYTE

This function converts an INT variable to a BYTE variable.

Instruction	Name	FB/FUN	Graphic expression
INT_TO_BYTE	Convert INT to BYTE	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	INT#0 to 255	INT#0
Out	Conversion result	Output	Conversion result	BYTE#16#00 to FF	BYTE#16#00

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In									OK	OK				
Out			OK	OK										

Function

This function converts INT data *In* to BYTE data *Out*.

Additional Information

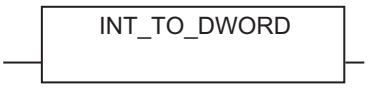
To convert BYTE data to INT data, refer to BYTE_TO_INT.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.
- If the input value is not INT#0 to INT#255, a cast error will occur and the program will stop.

INT_TO_DWORD

This function converts an INT variable to a DWORD variable.

Instruction	Name	FB/FUN	Graphic expression
INT_TO_DWORD	Convert INT to DWORD	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	INT#–32768 to 32767	INT#0
Out	Conversion result	Output	Conversion result	DWORD#16#00000000 to FFFFFFFFF	DWORD#16#00000000

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In									OK	OK				
Out							OK	OK						

Function

This function converts INT data *In* to DWORD data *Out*.

Example When Value of *In* Is Positive (INT#0 to INT#32767)

- The value of *Out* is DWORD#16#00000000 to DWORD#16#00007FFF according to the value of *In* (INT#0 to INT#32767).

Example When Value of *In* Is Negative (INT#–32768 to INT#–1)

- If the value of *In* is INT#–32768 (1111 1111 1111 1111 1000 0000 0000 0000 binary), the value of *Out* is DWORD#16#FFFF8000.
- If the value of *In* is INT#–1 (1111 1111 1111 1111 1111 1111 1111 1111 binary), the value of *Out* is DWORD#16#FFFFFFFF.

Additional Information

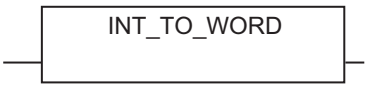
There is no instruction that converts DWORD data to INT data.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

INT_TO_WORD

This function converts an INT variable to a WORD variable.

Instruction	Name	FB/FUN	Graphic expression
INT_TO_WORD	Convert INT to WORD	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	INT#–32768 to 32767	INT#0
Out	Conversion result	Output	Conversion result	WORD#16#0000 to FFFF	WORD#16#0000

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In									OK	OK				
Out					OK	OK								

Function

This function converts INT data *In* to WORD data *Out*.

Example When Value of *In* Is Positive (INT#0 to INT#32767)

- The value of *Out* is WORD#16#0000 to WORD#16#7FFF according to the value of *In* (INT#0 to INT#32767).

Example When Value of *In* Is Negative (INT#–32768 to INT#–1)

- If the value of *In* is INT#–32768 (1000 0000 0000 0000 binary), the value of *Out* is WORD#16#8000.
- If the value of *In* is INT#–1 (1111 1111 1111 1111 binary), the value of *Out* is WORD#16#FFFF.

Additional Information

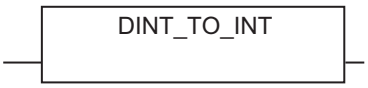
To convert WORD data to INT data, refer to WORD_TO_INT.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

DINT_TO_INT

This function converts a DINT variable to an INT variable.

Instruction	Name	FB/FUN	Graphic expression
DINT_TO_INT	Convert DINT to INT	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	DINT#–32768 to 32767	DINT#0
Out	Conversion result	Output	Conversion result	INT#–32768 to 32767	INT#0

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In											OK	OK		
Out									OK	OK				

Function

This function converts DINT data *In* to INT data *Out*.

Additional Information

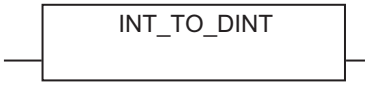
To convert INT data to DINT data, refer to INT_TO_DINT.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.
- If the input value is not DINT#–32768 to DINT#32767, a cast error will occur and the program will stop.

INT_TO_DINT

This function converts an INT variable to a DINT variable.

Instruction	Name	FB/FUN	Graphic expression
INT_TO_DINT	Convert INT to DINT	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	INT#–32768 to 32767	INT#0
Out	Conversion result	Output	Conversion result	DINT#–32768 to #32767	DINT#0

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In									OK	OK				
Out											OK	OK		

Function

This function converts INT data *In* to DINT data *Out*.

Example When Value of *In* Is INT#–32768 to INT#32767

- The value of *Out* will be DINT#–32768 to DINT#32767.

Additional Information

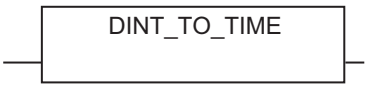
To convert DINT data to INT data, refer to DINT_TO_INT.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

DINT_TO_TIME

This function converts a DINT variable to a TIME variable.

Instruction	Name	FB/FUN	Graphic expression
DINT_TO_TIME	Convert DINT to TIME	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Integer	DINT#0 to 2147483647	DINT#0
Out	Conversion result	Output	Duration	T#0ms to T#24d20h31m23s647ms	T#0ms

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In											OK	OK		
Out													OK	OK

Function

This function converts DINT data *In* to TIME data *Out.*

Example When Value of *In* Is DINT#90090090

- The value of *Out* is 1 day 1 h 1 min 30 s 90 ms (T#1d1h1m30s090ms).

Additional Information

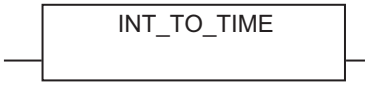
To convert TIME data to DINT data, refer to TIME_TO_DINT.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.
- If the input value is not DINT#0 to DINT#2147483647, a cast error will occur and the program will stop.

INT_TO_TIME

This function converts an INT variable to a TIME variable.

Instruction	Name	FB/FUN	Graphic expression
INT_TO_TIME	Convert INT to TIME	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	INT#0 to 32767	INT#0
Out	Conversion result	Output	Duration	T#0ms to T#32s767ms	T#0ms

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In									OK	OK				
Out													OK	OK

Function

This function converts INT data *In* to TIME data *Out*.

Example When Value of *In* Is INT#10500

- The value of *Out* is 10 s 500 ms (T#10s500ms).

Additional Information

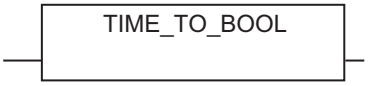
To convert TIME data to INT data, refer to TIME_TO_INT.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.
- If the input value is not INT#0 to INT#32767, a cast error will occur and the program will stop.

TIME_TO_BOOL

This function converts a TIME variable to a BOOL variable.

Instruction	Name	FB/FUN	Graphic expression
TIME_TO_BOOL	Convert TIME to BOOL	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Duration	T#0ms to T#49d17h2m47s295ms	T#0ms
Out	Conversion result	Output	Conversion result	TRUE or FALSE	FALSE

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In													OK	OK
Out	OK	OK												

Function

This function converts TIME data *In* to BOOL data *Out*.

If the value of *In* is 0 ms (T#0ms), the value of *Out* is FALSE.

If the value of *In* is 1 ms (T#1ms) to 49 days 17 h 2 min 47 s 295 ms (T#49d17h2m47s295ms) (i.e., not T#0ms), the value of *Out* is TRUE.

Additional Information

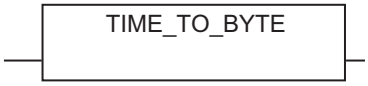
To convert BOOL data to TIME data, refer to BOOL_TO_TIME.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

TIME_TO_BYTE

This function converts a TIME variable to a BYTE variable.

Instruction	Name	FB/FUN	Graphic expression
TIME_TO_BYTE	Convert TIME to BYTE	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Duration	T#0ms to T#255ms	T#0ms
Out	Conversion result	Output	Conversion result	BYTE#16#00 to FF	BYTE#16#00

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In													OK	OK
Out			OK	OK										

Function

This function converts TIME data *In* to BYTE data *Out*.

Additional Information

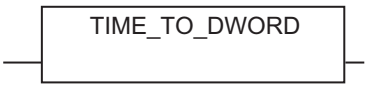
To convert BYTE data to TIME data, refer to BYTE_TO_TIME.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.
- If the input value is not TIME#0ms to TIME#255ms, a cast error will occur and the program will stop.

TIME_TO_DWORD

This function converts a TIME variable to a DWORD variable.

Instruction	Name	FB/FUN	Graphic expression
TIME_TO_DWORD	Convert TIME to DWORD	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Duration	T#0ms to T#49d17h2m47s295ms	T#0ms
Out	Conversion result	Output	Conversion result	DWORD#16#00000000 to FFFFFFFF	DWORD#16#00000000

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In													OK	OK
Out							OK	OK						

Function

This function converts TIME data *In* to DWORD data *Out*.

Example When Value of *In* Is 49 days 17 h 2 min 47 s 295 ms (T#49d17h2m47s295ms)

- The value of *Out* will be DWORD#16#FFFFFFFF.

Additional Information

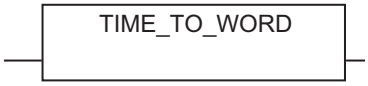
To convert DWORD data to TIME data, refer to DWORD_TO_TIME.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

TIME_TO_WORD

This function converts a TIME variable to a WORD variable.

Instruction	Name	FB/FUN	Graphic expression
TIME_TO_WORD	Convert TIME to WORD	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Duration	T#0ms to T#65s535ms	T#0ms
Out	Conversion result	Output	Conversion result	WORD#16#0000 to FFFF	WORD#16#0000

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In													OK	OK
Out					OK	OK								

Function

This function converts TIME data *In* to WORD data *Out*.

Example When Value of *In* Is 50 s 505 ms (T#50s505ms)

- The value of *Out* will be WORD#16#C549.

Additional Information

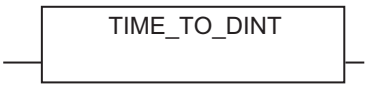
To convert WORD data to TIME data, refer to WORD_TO_TIME.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.
- If the input value is not TIME#0ms to TIME#65s535ms, a cast error will occur and the program will stop.

TIME_TO_DINT

This function converts a TIME variable to a DINT variable.

Instruction	Name	FB/FUN	Graphic expression
TIME_TO_DINT	Convert TIME to DINT	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Duration	T#0ms to T#49d17h2m47s295ms	T#0ms
Out	Conversion result	Output	Integer	DINT#0 to 2147483647	DINT#0

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In													OK	OK
Out											OK	OK		

Function

This function converts TIME data *In* to DINT data *Out*.

Example When Value of *In* Is 1 day 1 h 1 min 30 s 90 ms (T#1d1h1m30s090ms)

- The value of *Out* will be DINT#90090090.

Additional Information

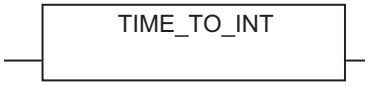
To convert DINT data to TIME data, refer to DINT_TO_TIME.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

TIME_TO_INT

This function converts a TIME variable to an INT variable.

Instruction	Name	FB/FUN	Graphic expression
TIME_TO_INT	Convert TIME to INT	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Duration	T#0ms to T#32s767ms	T#0ms
Out	Conversion result	Output	Conversion result	INT#0 to 32767	INT#0

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In													OK	OK
Out									OK	OK				

Function

This function converts TIME data *In* to INT data *Out*.

Example When Value of *In* Is 10 s 500 ms (T#10s500ms)

- The value of *Out* will be INT#10500.

Additional Information

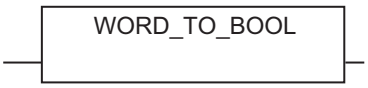
To convert INT data to TIME data, refer to INT_TO_TIME.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.
- If the input value is not TIME#0ms to TIME#32s767ms, a cast error will occur and the program will stop.

WORD_TO_BOOL

This function converts a WORD variable to a BOOL variable.

Instruction	Name	FB/FUN	Graphic expression
WORD_TO_BOOL	Convert WORD to BOOL	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to convert	Input	Data to convert	WORD#16#0000 to FFFF	WORD#16#0000
Out	Conversion result	Output	Conversion result	TRUE or FALSE	FALSE

If you omit an input or output parameter, a building error will occur.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In					OK	OK								
Out	OK	OK												

Function

This function converts WORD data *In* to BOOL data *Out*.

If the value of *In* is WORD#16#0000, the value of *Out* is FALSE.

If the value of *In* is WORD#16#0001 to WORD#16#FFFF (i.e., not WORD#16#0000), the value of *Out* is TRUE.

Additional Information

To convert BOOL data to WORD data, refer to BOOL_TO_WORD.

Precautions for Correct Use

- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.
- If the input value is not WORD#0 or WORD#1, a cast error will occur and the program will stop.

Boolean Operation Instructions

Type	Instruction	Name	Description	Page
Boolean operations	AND	Logical AND	Performs a logical AND on multiple Boolean variables.	P. 2-52
	OR	Logical OR	Performs a logical OR on multiple Boolean variables.	P. 2-52
	XOR	Exclusive logical OR	Performs an exclusive logical OR on multiple Boolean variables.	P. 2-52
	NOT	Bit Reversal	Reverses the value of a Boolean variable.	P. 2-54

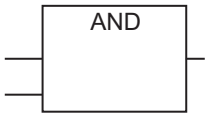
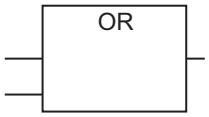
AND, OR, and XOR

These instructions perform Boolean operations.

AND: Logical AND

OR: Logical OR

XOR: Exclusive logical OR

Instruction	Name	FB/FUN	Graphic expression
AND	Logical AND	FUN	
OR	Logical OR	FUN	
XOR	Exclusive logical OR	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In1 to InN	Data to process	Input	Data to process	TRUE or FALSE	FALSE
Out	Processing result	Output	Processing result	TRUE or FALSE	---

If you omit an input or output parameter, a building error will occur.
An error will not occur if the output terminal is not used or if it is connected to an input terminal on the next instruction.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In1 to InN	OK	OK												
Out	OK	OK												

Function

These instructions perform operations for two or more Boolean variables *In1* to *InN*.

The relationships between input and output variables are given in the following tables.

AND: If all inputs are TRUE, then the processing result is TRUE. Otherwise, the processing result is FALSE.

***In1* bit**	***In2* bit**		***InN* bit**	***Out* bit**
FALSE	FALSE	· · ·	FALSE	FALSE
FALSE	FALSE	· · ·	TRUE	FALSE
FALSE	TRUE	· · ·	TRUE	FALSE
TRUE	FALSE	· · ·	FALSE	FALSE
TRUE	FALSE	· · ·	TRUE	FALSE
TRUE	TRUE	· · ·	TRUE	TRUE

OR: If all inputs are FALSE, then the processing result is FALSE. Otherwise, the processing result is TRUE.

***In1* bit**	***In2* bit**		***InN* bit**	***Out* bit**
FALSE	FALSE	· · ·	FALSE	FALSE
FALSE	FALSE	· · ·	TRUE	TRUE
FALSE	TRUE	· · ·	TRUE	TRUE
TRUE	FALSE	· · ·	FALSE	TRUE
TRUE	FALSE	· · ·	TRUE	TRUE
TRUE	TRUE	· · ·	TRUE	TRUE

XOR: If both inputs have the same value, then the processing result is FALSE. If one bit is TRUE and the other is FALSE, then the processing result is TRUE.

***In1* bit**	***In2* bit**	***Out* bit**
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

Additional Information

With AND and OR, you can perform an operation for two or more variables, *In1* to *InN*, at the same time. With XOR, however, you can perform an operation for only two variables, *In1* and *In2*, at the same time. A building error will occur if there are three or more input terminals for XOR.

Precautions for Correct Use

- You must use Boolean variables for *In1* to *InN* and for *Out*.
- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.

  Setting a Safety Data Type Variable for the Output Terminal

  AND: Set a safety data type variable for at least one of the input terminals.

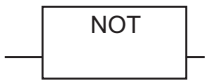
  OR/XOR: Set safety data type variables for all of the input terminals.

  Setting a Standard Data Type Variable for the Output Terminal

  AND/OR/XOR: Use either safety data type variable or standard data type variable for the input terminals.

NOT

This function reverses the value of a Boolean bit.

Instruction	Name	FB/FUN	Graphic expression
NOT	Bit Reversal	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In	Data to process	Input	Data to process	TRUE or FALSE	---
Out	Processing result	Output	Processing result	TRUE or FALSE	FALSE

If you omit an input or output parameter, a building error will occur.
An error will not occur if the output terminal is not used or if it is connected to an input terminal on the next instruction.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	OK	OK												
Out	OK	OK												

Function

This function reverses the TRUE/FALSE value of the Boolean variable *In*.
The relationships between input and output variables are given in the following table.

In bit	*Out* bit
FALSE	TRUE
TRUE	FALSE

Precautions for Correct Use

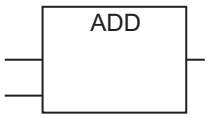
- You must use Boolean variables for *In* and *Out*.
- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the input terminal as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

Math Instructions

Type	Instruction	Name	Description	Page
Math	ADD	Addition	Adds integers or durations.	P. 2-56
	SUB	Subtraction	Subtracts integers or durations.	P. 2-58
	MUL	Multiplication	Multiplies integers or a duration.	P. 2-60
	DIV	Division	Divides integers or a duration.	P. 2-62

ADD

This function adds integers or durations.

Instruction	Name	FB/FUN	Graphic expression
ADD	Addition	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In1 to InN	Values to add	Input	Values to add	Depends on data type.	---
Out	Output value	Output	Output value	Depends on data type.	---

If you omit an input or output parameter, a building error will occur.
An error will not occur if the output terminal is not used or if it is connected to an input terminal on the next instruction.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In1 to InN									OK	OK	OK	OK	OK	OK
Out									OK	OK	OK	OK	OK	OK

Function

This function adds integers or durations, and outputs the result to output value *Out*.

An overflow occurs if the sum of *In1* to *InN* exceeds the valid range of the data type of the addition result. If an overflow occurs, the data types of *In1* to *InN*, the data type of the addition result, and the value of the addition result will be as shown in the following table.

I/O data types	Value of addition results
Integers	Of the sum of *In1* to *InN*, the addition result will be the value that can be expressed by the number of bits in the data type of the addition result.*1*2
Durations	Of the sum of *In1* to *InN*, the addition result will be the value that can be expressed with DWORD data.*3

*1. For example, if the value of *In1* is INT#32767 and the value of *In2* is INT#3, the addition result will be 32770. If an INT variable is set for the output, the value of the addition result will be the value that can be expressed with lower 16 bits of the sum (32,770), i.e., –32,766. If a DINT variable is set for the output, the addition result is DINT#32770.

*2. If the result exceeds the valid range of DINT data, the result will be the value that can be expressed with the lower 32 bits.

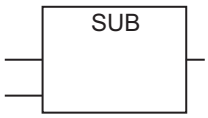
*3. For example, if the value of *In1* is TIME#49d17h2m47s295ms and the value of *In2* is TIME#5ms, the value of the addition result is TIME#49d17h2m47s30ms. However, the maximum value of TIME is the same as for DWORD (4294967295), so the addition result will be the value that can be expressed with 32 bits, i.e., TIME#4ms.

Precautions for Correct Use

- The data types of *In1* to *InN* and *Out* can be different. If they are different, calculations are performed with the data type that includes the range of all of the data types. For example, if *In0* is INT data and *In1* is DINT data, calculations are performed with DINT data. Therefore, addition result *Out* will be DINT data.
- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set safety data type variables for all of the input terminals.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

SUB

This function subtracts integers or durations.

Instruction	Name	FB/FUN	Graphic expression
SUB	Subtraction	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In1	Minuend	Input	Minuend	Depends on data type.	---
In2	Subtrahend	Input	Subtrahend	Depends on data type.	---
Out	Output value	Output	Output value	Depends on data type.	---

If you omit an input or output parameter, a building error will occur.
An error will not occur if the output terminal is not used or if it is connected to an input terminal on the next instruction.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In1									OK	OK	OK	OK	OK	OK
In2									OK	OK	OK	OK	OK	OK
Out									OK	OK	OK	OK	OK	OK

Function

This function subtracts subtrahend *In2* from minuend *In1* and outputs the result to output value *Out*.

An overflow occurs if the difference between *In1* and *In2* exceeds the valid range of the data type of the subtraction result.
If an overflow occurs, the data types of *In1* and *In2*, the data type of the subtraction result, and the value of the subtraction result will be as shown in the following table.

I/O data types	Value of subtraction results
Integers	Of the difference between *In1* and *In2*, the subtraction result will be the value that can be expressed by the number of bits in the data type of the subtraction result.*1*2
Durations	Of the difference between *In1* and *In2*, the subtraction result will be the value that can be expressed by DWORD data.*3

*1. For example, if the value of *In1* is INT#–5 and the value of In2 is INT#32767, the subtraction result will be –32772. If an INT variable is set for the output, the value of the subtraction result will be the value that can be expressed with lower 16 bits of the difference (–32772), i.e., 32764. If a DINT variable is set for the output, the subtraction results is DINT#–32772.

*2. If the result exceeds the valid range of DINT data, the result will be the value that can be expressed with the lower 32 bits.

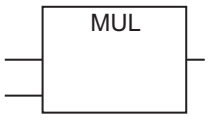
*3. For example, if the value of *In1* is TIME#10ms and the value of *In2* is TIME#14ms, the value of the subtraction result is TIME#−4ms. However, the maximum value of TIME is the same as for DWORD (4294967295), so the subtraction result will be the value that can be expressed with 32 bits, i.e., T#49d17h2m47s292ms. Although negative time does not actually exist, the value is expressed as a negative value.

Precautions for Correct Use

- The data types of *In1*, *In2*, and *Out* can be different. If they are different, calculations are performed with the data type that includes the range of all of the data types. For example, if *In0* is INT data and *In1* is DINT data, calculations are performed with DINT data. Therefore, subtraction result *Out* will be DINT data.
- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the two input terminals as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

MUL

This function multiplies integers or a duration.

Instruction	Name	FB/FUN	Graphic expression
MUL	Multiplication	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In1 to InN	Values to multiply	Input	Values to multiply	Depends on data type.	---
Out	Output value	Output	Output value	Depends on data type.	---

If you omit an input or output parameter, a building error will occur.
An error will not occur if the output terminal is not used or if it is connected to an input terminal on the next instruction.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In1									OK	OK	OK	OK	OK	OK
In2 to InN									OK	OK	OK	OK		
Out									OK	OK	OK	OK	OK	OK

Function

This function multiplies multiply values *In1* to *InN* and outputs the result to output value *Out.*

An overflow occurs if the product of *In1* to *InN* exceeds the valid range of the data type of the multiplication result. If an overflow occurs, the data types of *In1* to *InN*, the data type of the multiplication result, and the value of the multiplication result will be as shown in the following table.

I/O data types	Value of multiplication results
Integers	Of the product of *In1* to *InN*, the multiplication result will be the value that can be expressed by the number of bits in the data type of the addition result.*1*2
Durations	Of the product of In1 to InN, the multiplication result will be the value that can be expressed with DWORD data.*3

*1. For example, if the value of *In1* is INT#16390 and the value of *In2* is INT#2, the multiplication result will be 32780. If an INT variable is set for the output, the value of the multiplication result will be the value that can be expressed with lower 16 bits of the product (32,780), i.e., –32,756. If a DINT variable is set for the output, the addition result is DINT#32780.

*2. If the result exceeds the valid range of DINT data, the result will be the value that can be expressed with the lower 32 bits.

*3. For example, if the value of *In1* is T#24d20h31m23s649ms and the value of *In2* is INT#2, the value of the multiplication result is T#49d17h2m47s298ms. However, the maximum value of TIME is the same as for DWORD (4294967295), so the multiplication result will be the value that can be expressed with 32 bits, i.e., T#2ms.

Additional Information

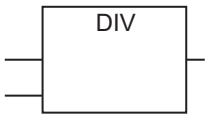
You cannot include more than one duration in the values to multiply.
To multiply a duration, set a duration for *In1* and *Out* and set integers for *In2* to *InN*.

Precautions for Correct Use

- When multiplying integers, the data types of *In1* to *InN* and *Out* can be different. If they are different, calculations are performed with the data type that includes the range of all of the data types. For example, if *In1* is INT data and *In2* is DINT data, calculations are performed with DINT data. Therefore, multiplication result *Out* will be DINT data.
- If you use duration data for the data to multiply, use duration data for one of *In1* to *InN* and for *Out*.
- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur. If you set a safety data type variable for the output terminal, set safety data type variables for all of the input terminals.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.
- If you set a safety data type variable for the output terminal, set safety data type variables for all of the input terminals.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

DIV

This function divides integers or a duration.

Instruction	Name	FB/FUN	Graphic expression
DIV	Division	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In1	Dividend	Input	Dividend	Depends on data type.	---
In2	Divisor	Input	Divisor	Depends on data type.	---
Out	Output value	Output	Output value	Depends on data type.	---

If you omit an input or output parameter, a building error will occur.
An error will not occur if the output terminal is not used or if it is connected to an input terminal on the next instruction.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In1									OK	OK	OK	OK	OK	OK
In2									OK	OK	OK	OK		
Out									OK	OK	OK	OK	OK	OK

Function

This function divides dividend *In1* by divisor *In2* and outputs the result to output value *Out*.

Any remainder is truncated.

Additional Information

You cannot include more than one duration in the values to divide.

To divide a duration, set a duration for *In1* and *Out* and set an integer for *In2*.

Do not allow the divisor to equal 0.

If the divisor is 0, a Division by Zero error will occur and the program will stop.

Precautions for Correct Use

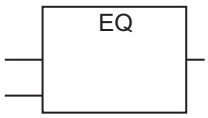
- The data types of *In1*, *In2*, and *Out* can be different. If they are different, calculations are performed with the data type that includes the range of all of the data types. For example, if *In1* is INT data and *In2* is DINT data, calculations are performed with DINT data. Therefore, division result *Out* will be DINT data.
- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the two input terminals as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

Comparison Instructions

Type	Instruction	Name	Description	Page
Comparison	EQ	Equal	Determines if the values of two variables are equivalent.	P. 2-66
	NE	Not Equal	Determines if the values of two variables are not equivalent.	P. 2-67
	LT	Less Than	Performs a less than comparison between two values.	P. 2-68
	LE	Less Than Or Equal	Performs a less than or equal comparison between two values.	P. 2-68
	GT	Greater Than	Performs a greater than comparison between two values.	P. 2-68
	GE	Greater Than Or Equal	Performs a greater than or equal comparison between two values.	P. 2-68

EQ

This function determines if the values of two variables are equivalent.

Instruction	Name	FB/FUN	Graphic expression
EQ	Equal	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In1 or In2	Comparison data	Input	Values to compare	Depends on data type.	---
Out	Comparison result	Output	Comparison result	Depends on data type.	---

If you omit an input or output parameter, a building error will occur.
An error will not occur if the output terminal is not used or if it is connected to an input terminal on the next instruction.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In1 or In2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
Out	OK	OK												

Function

This function determines if the values of two variables *In1* and *In2* are equivalent.

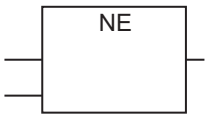
If they are equivalent, comparison result *Out* changes to TRUE. Otherwise, the value of *Out* is FALSE.

Precautions for Correct Use

- You can compare *In1* and *In2* even if they have different data types, such as a safety data type and a standard data type, as long as the notations and sizes of the data types are the same. You cannot compare data with data types that have different notations or sizes, such as WORD and INT. You can compare integer data, such as INT data with SAFEINT data and DINT data with SAFEDINT data.
- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the two input terminals as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

NE

This function determines if the values of two variables are not equivalent.

Instruction	Name	FB/FUN	Graphic expression
NE	Not Equal	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In1 or In2	Comparison data	Input	Values to compare	Depends on data type.	---
Out	Comparison result	Output	Comparison result	Depends on data type.	---

If you omit an input or output parameter, a building error will occur.
An error will not occur if the output terminal is not used or if it is connected to an input terminal on the next instruction.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In1 or In2	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
Out	OK	OK												

Function

This function determines if the values of two variables *In1* and *In2* are not equivalent.

If they are not equivalent, the comparison result *Out* is TRUE. If they are equivalent, *Out* is FALSE.

Precautions for Correct Use

- You can compare *In1* and *In2* even if they have different data types, such as a safety data type and a standard data type, as long as the notations and sizes of the data types are the same. You cannot compare data with data types that have different notations or sizes, such as WORD and INT. You can compare integer data, such as INT data with SAFEINT data and DINT data with SAFEDINT data.
- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the two input terminals as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

LT, LE, GT, and GE

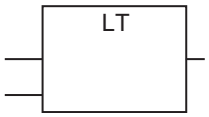
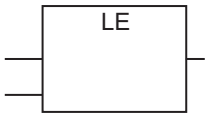
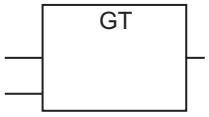
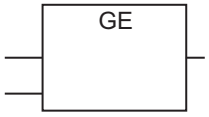
These instructions compare the sizes of two values.

LT: Performs a less than comparison between two values.

LE: Performs a less than or equal comparison between two values.

GT: Performs a greater than comparison between two values.

GE: Performs a greater than or equal comparison between two values.

Instruction	Name	FB/FUN	Graphic expression
LT	Less Than	FUN	
LE	Less Than Or Equal	FUN	
GT	Greater Than	FUN	
GE	Greater Than Or Equal	FUN	

Variables

	Name	I/O	Description	Valid range	Default
In1 or In2	Comparison data	Input	Value to compare	Depends on data type.	---
Out	Comparison result	Output	Comparison result	Depends on data type.	---

If you omit an input or output parameter, a building error will occur.
An error will not occur if the output terminal is not used or if it is connected to an input terminal on the next instruction.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In1 or In2									OK	OK	OK	OK	OK	OK
Out	OK	OK												

Function

These functions compare the values of two variables, *In1* and *In2*.

The output value *Out* is shown below for each instruction.

LT: If *In1* is less than *In2*, the result is TRUE. Otherwise the result is FALSE.

LE: If *In1* is less than or equal to *In2*, the result is TRUE. Otherwise the result is FALSE.

GT: If *In1* is greater than *In2*, the result is TRUE. Otherwise the result is FALSE.

GE: If *In1* is greater than or equal to *In2*, the result is TRUE. Otherwise the result is FALSE.

Additional Information

The relationship between values with data types that are integers or durations are determined as given in the following table.

Data types	Relationship
INT, SAFEINT, DINT, or SAFEDINT	The sign is included in the comparison.
TIME or SAFETIME	The values of the days, hours, minutes, seconds, and milliseconds are compared.

Precautions for Correct Use

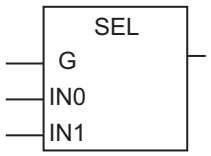
- You can compare *In1* and *In2* even if they have different data types, such as a safety data type and a standard data type, as long as the notations and sizes of the data types are the same. You cannot compare data with data types that have different notations or sizes, such as WORD and INT. You can compare integer data, such as INT data with SAFEINT data and DINT data with SAFEDINT data.
- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur. If you set a safety data type variable for the output terminal, set a safety data type variable for the two input terminals as well.
  If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

Selection Instructions

Type	Instruction	Name	Description	Page
Selection	SEL	Bit Selection	Selects one of two selections.	P. 2-72
	MUX	Multiplexer	Selects one of multiple selections.	P. 2-74

SEL

This function selects one of two selections.

Instruction	Name	FB/FUN	Graphic expression
SEL	Bit Selection	FUN	

Variables

	Name	I/O	Description	Valid range	Default
G	Gate	Input	FALSE: Selects *In0*. TRUE: Selects *In1*.	Depends on data type.	FALSE
In0 or In1	Selections		Selections		---
Out	Selection result	Output	Selection result	Depends on data type.	---

If you omit an input or output parameter, a building error will occur.
An error will not occur if the output terminal is not used or if it is connected to an input terminal on the next instruction.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
G	OK	OK												
In0 or In1	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
Out	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK

Function

This function specifies one of two selections, *In0* and *In1*.

Use gate *G* to specify which of *In0* and *In1* to select.

If *G* is FALSE, *In0* is assigned to *Out*. If *G* is TRUE, *In1* is assigned to *Out*.

Additional Information

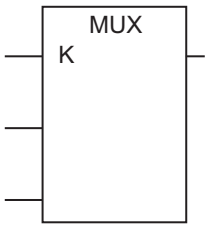
Use the MUX instruction to select one of two or more selections.

Precautions for Correct Use

- The data types of *In0*, *In1*, and *Out* can be different. If they are different, calculations are performed with the data type that includes the range of all of the data types. For example, if *In0* is INT data and *In1* is DINT data, calculations are performed with DINT data. Therefore, selection result *Out* will be DINT data.
- The input condition depends on whether the output is safety data or standard data. If the condition is not met, a building error will occur.
- If you set a safety data type variable for the output terminal, set a safety data type variable for the two input terminals as well.
- If you set a standard data type variable for the output terminal, you can set either a safety data type variable or a standard data type variable for the input terminal.

MUX

This function selects one of multiple selections.

Instruction	Name	FB/FUN	Graphic expression
MUX	Multiplexer	FUN	

Variables

	Name	I/O	Description	Valid range	Default
K	Selector	Input	0: Selects *In0*. 1: Selects *In1*. 2: Selects *In2*. N: Selects *InN*.	Depends on data type.	---
In0 to InN	Selections		Selections		
Out	Selection result	Output	Selection result	Depends on data type.	---

If you omit an input or output parameter, a building error will occur.
An error will not occur if the output terminal is not used or if it is connected to an input terminal on the next instruction.

	Boolean		Bit strings						Integers				Durations	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
K									OK	OK	OK	OK		
In0 to InN	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
Out	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK

Function

This function selects one of two to N selections, *In0* to *InN*.

Selector *K* specifies which of *In0* to *InN* to select.

The value of one of the input variables is assigned to *Out* according to the value of *K*. *In0* is assigned if *K* is 0, *InN* is assigned if *K* is N.

Additional Information

Use the SEL instruction to select one of two selections.

Precautions for Correct Use

- *In0*, *In1*, and *Out* may have different data types, but observe the following precautions.
  - Set the valid range of *Out* to include the valid ranges of *In0* to *InN*.
  - If the value of *K* is outside the valid range (i.e., less than 0 or greater than N), an MUX Error will occur and the program will stop.

3

Safety Standard Function Blocks

This section gives the specifications of the safety standard function blocks that you can use for NX-series safety control.

Safety Standard Function Block Instructions

Instruction	Variable	Function	Page
SF_CTD	Down-counter	Decrements the counter value when the counter input signal is received.	P. 3-3
SF_CTU	Up-counter	Increments the counter value when the counter input signal is received.	P. 3-5
SF_CTUD	Up-down Counter	Creates an up-down counter that operates according to an up-counter input and a down-counter input.	P. 3-7
SF_F_TRIG	Down Trigger	Outputs TRUE for one task period only when the input signal changes to FALSE.	P. 3-10
SF_R_TRIG	Up Trigger	Outputs TRUE for one task period only when the input signal changes to TRUE.	P. 3-11
SF_RS	Reset-Priority Keep	Retains the value of a SAFEBOOL variable.	P. 3-12
SF_SR	Set-Priority Keep	Retains the value of a SAFEBOOL variable.	P. 3-13
SF_TOF	Off-Delay Timer	Outputs FALSE when the set time elapses after the timer starts.	P. 3-14
SF_TON	On-Delay Timer	Outputs TRUE when the set time elapses after the timer starts.	P. 3-16
SF_TP	Timer Pulse	Outputs TRUE during the set time after the timer starts.	P. 3-18

SF_CTD

This FB decrements the counter value when the counter input signal is received.

Instruction	Name	FB/FUN	Graphic expression
SF_CTD	Down-counter	FB	

Variables

Input Variables

Variable	Name	Data type	Valid range	Default	Description
CD	Counter input	BOOL	TRUE or FALSE	FALSE	Counter input
LOAD	Load signal	BOOL	TRUE or FALSE	FALSE	TRUE: Set CV to PV.
PV	Preset value	SAFEINT	0 to 32767	0	Counter preset value

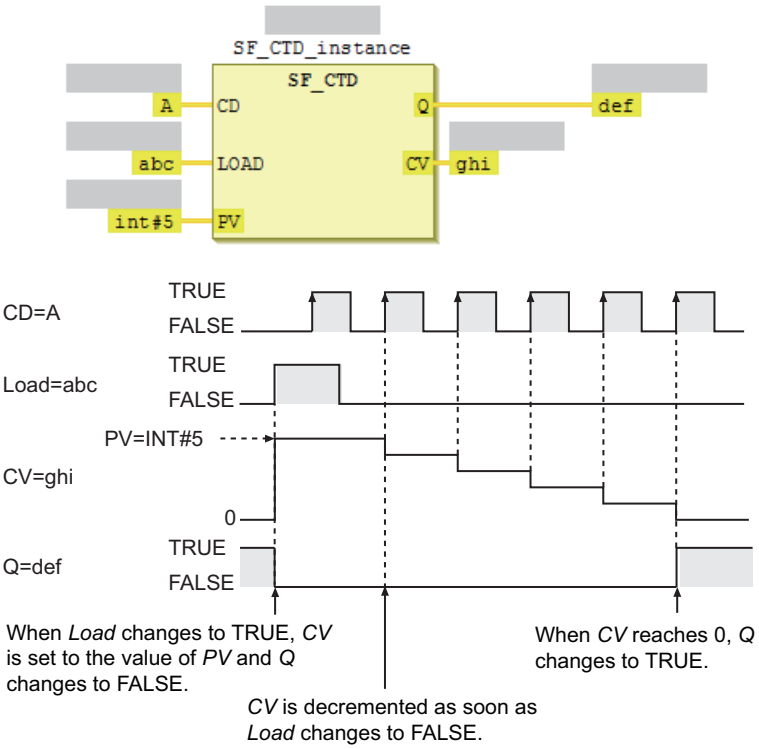
Output Variables

Variable	Name	Data type	Valid range	Default	Description
Q	Counter output	SAFEBOOL	TRUE or FALSE	FALSE	TRUE: *CV* is 0 or lower. FALSE: *CV* is 1 or higher.
CV	Counter value	SAFEINT	0 to 32767	0	Counter present value

Function

- The SF_CTD instruction creates a down counter. The preset value and counter value must have a SAFEINT data type.
- When load signal *Load* changes to TRUE, counter value *CV* is set to the value of preset value *PV* and counter output *Q* changes to FALSE.
- When counter input signal *CD* changes to TRUE, *CV* is decremented.
- When the value of *CV* reaches 0 or less, the value of *Q* changes to TRUE.
- After the value of *CV* reaches 0 or less, *CV* does not change even if *CD* changes to TRUE.
- *CD* is ignored while *Load* is TRUE. *CV* is not decremented.

The following figure shows a programming example and timing chart for a *PV* of INT#5.

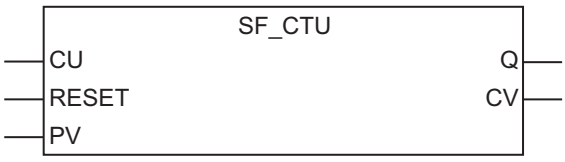


Additional Information

- Use the SF_CTU instruction (P. 3-5) to create a counter that increments the counter value each time the counter input signal is received.
- Use the SF_CTUD instruction (P. 3-7) to create a counter that is both incremented and decremented.

SF_CTU

This FB increments the counter value when the counter input signal is received.

Instruction	Name	FB/FUN	Graphic expression
SF_CTU	Up-counter	FB	

Variables

Input Variables

Variable	Name	Data type	Valid range	Default	Description
CU	Counter input	BOOL	TRUE or FALSE	FALSE	Counter input
RESET	Reset signal	BOOL	TRUE or FALSE	FALSE	TRUE: Reset *CV* to 0.
PV	Preset value	SAFEINT	0 to 32767	0	Counter preset value

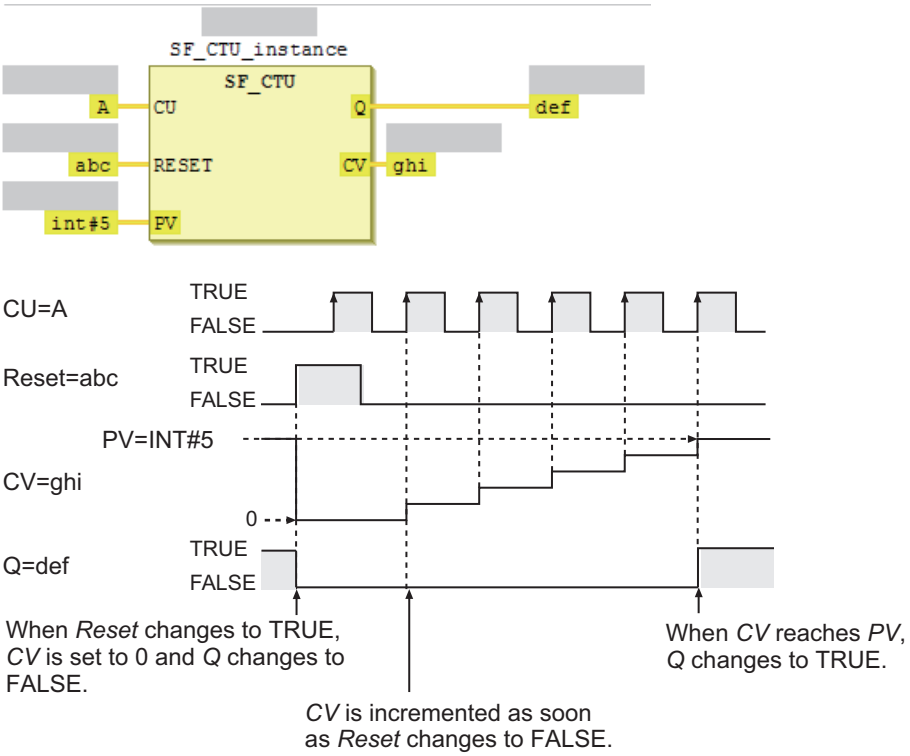
Output Variables

Variable	Name	Data type	Valid range	Default	Description
Q	Counter output	SAFEBOOL	TRUE or FALSE	FALSE	TRUE: *CV* is greater than or equal to *PV*. FALSE: *CV* is less than *PV*.
CV	Counter value	SAFEINT	0 to 32767	0	Counter present value

Function

- This FB creates an up counter. The preset value and counter value must have a SAFEINT data type.
- When reset signal *RESET* changes to TRUE, counter value *CV* changes to 0 and counter output *Q* changes to FALSE.
- When counter input signal *CU* changes to TRUE, *CV* is incremented. When the value of *CV* reaches the value of *PV* or higher, the value of *Q* changes to TRUE.
- Even after the value of *CV* exceeds the value of *PV*, *CV* is incremented to up to 32,767 when *CU* changes to TRUE.
- *CU* is ignored while *RESET* is TRUE. *CV* is not incremented.

The following figure shows a programming example and timing chart for a *PV* of INT#5.

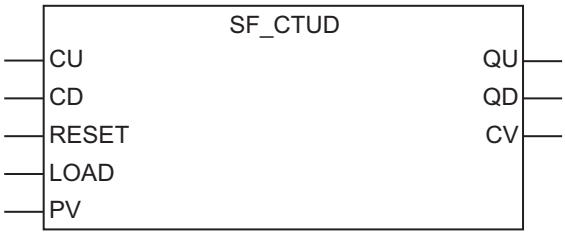


Additional Information

- Use the SF_CTD instruction to create a counter that decrements the counter value each time the counter input signal is received.
- Use the SF_CTUD instruction to create a counter that is both incremented and decremented.

SF_CTUD

This FB creates an up-down counter that operates according to an up-counter input and a down-counter input.

Instruction	Name	FB/FUN	Graphic expression
SF_CTUD	Up-down Counter	FB	

Variables

Input Variables

Variable	Name	Data type	Valid range	Default	Description
CU	Up counter input	BOOL	TRUE or FALSE	FALSE	Up counter input
CD	Down-counter input	BOOL	TRUE or FALSE	FALSE	Down-counter input
RESET	Reset signal	BOOL	TRUE or FALSE	FALSE	TRUE: Reset *CV* to 0.
LOAD	Load signal	BOOL	TRUE or FALSE	FALSE	TRUE: Set *CV* to *PV*.
PV	Preset value	SAFEINT	0 to 32767	0	This is the count-up value for an up counter or the initial value for a down counter.

Output Variables

Variable	Name	Data type	Valid range	Default	Description
QU	Up-counter output	SAFEBOOL	TRUE or FALSE	FALSE	TRUE: *CV* is greater than or equal to *PV*. FALSE: *CV* is less than *PV*.
QD	Down-counter output	SAFEBOOL	TRUE or FALSE	FALSE	TRUE: *CV* is 0 or lower. FALSE: *CV* is 1 or higher.
CV	Counter value	SAFEINT	0 to 32767	0	Counter present value

Function

- This FB creates an up-down counter that operates according to an up-counter input signal and a down-counter input signal.
- The counter has the functions of both an up counter and a down counter.
- The preset value and counter value must have a SAFEINT data type.

Operation as an Up Counter

When reset signal *RESET* changes to TRUE, counter value *CV* changes to 0 and up-counter output *QU* changes to FALSE.

When up-counter input signal *CU* changes to TRUE, *CV* is incremented. When the value of *CV* reaches the value of *PV* or higher, the value of *QU* changes to TRUE.

After the value of *CV* reaches the value of *PV* or higher, the value of *CV* does not change even if the value of *CU* changes to TRUE.

Operation as a Down Counter

When load signal *LOAD* changes to TRUE, counter value *CV* changes to the value of preset value *PV* and down-counter output *QD* changes to FALSE.

When down-counter input signal *CD* changes to TRUE, *CV* is decremented. When the value of *CV* reaches 0 or less, the value of *QD* changes to TRUE.

After the value of *CV* reaches 0 or less, *CV* does not change even if *CD* changes to TRUE.

Common Operation for Up and Down Counters

CU and *CD* are ignored while *LOAD* or *RESET* is TRUE. *CV* is not incremented or decremented.

If both *CU* and *CD* change to TRUE at the same time, *CV* will not change.

If *RESET* and *LOAD* are both TRUE, *RESET* has priority and the value of *CV* changes to 0.

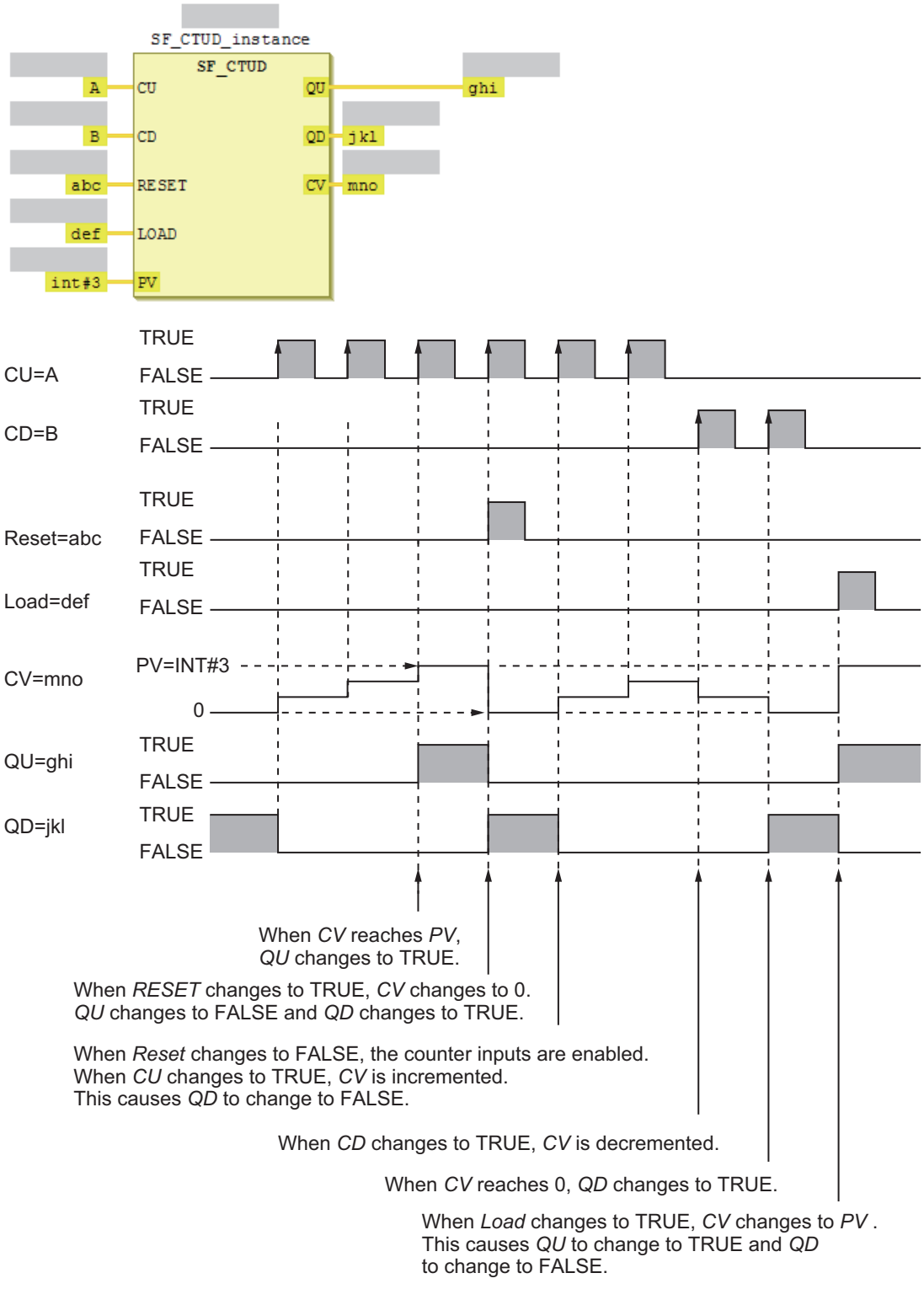
If *RESET* changes to TRUE, *CV* changes to 0, and so *QD* changes to TRUE.

If *LOAD* changes to TRUE, the value of *CV* changes to *PV*, and so *QU* changes to TRUE.

The following table shows the relationship between *RESET*, *LOAD*, *CV*, *QU*, and *QD*. This assumes that the value of *PV* is larger than 0.

RESET	Load	CV	QU	QD	Operation
FALSE	FALSE	0 or lower	FALSE	TRUE	Only an up counter operation is performed. *CV* is incremented when *CU* changes to TRUE. It is not decremented when *CD* changes to TRUE.
		Between 0 and *PV*	FALSE	FALSE	Both up and down counter operation is performed. *CV* is incremented when *CU* changes to TRUE and decremented when *CD* changes to TRUE.
		PV or higher	TRUE	FALSE	Only down counter operation is performed. *CV* is decremented when *CD* changes to TRUE. It is not incremented when *CU* changes to TRUE.
TRUE	FALSE	0	FALSE	TRUE	The up counter is reset. The value of *CV* is set to 0.
FALSE	TRUE	*PV*	TRUE	FALSE	The down counter is reset. The value of *CV* is set to *PV*.
TRUE	TRUE	0	FALSE	TRUE	The up counter is reset. *Reset* takes priority over *Load*. The value of *CV* is set to 0.

The following figure shows a programming example and timing chart for a *PV* of INT#3.

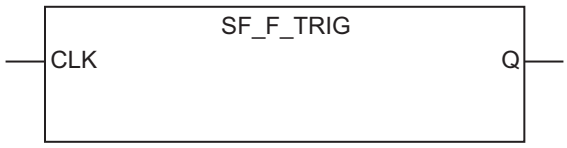


Additional Information

Use the SF_CTD instruction or SF_CTU instruction to create a counter that only decrements or only increments.

SF_F_TRIG

This FB outputs TRUE for one task period only when the input signal changes to FALSE.

Instruction	Name	FB/FUN	Graphic expression
SF_F_TRIG	Down Trigger	FB	

Variables

Input Variables

Variable	Name	Data type	Valid range	Default	Description
CLK	Input signal	SAFEBOOL	TRUE or FALSE	FALSE	Input signal

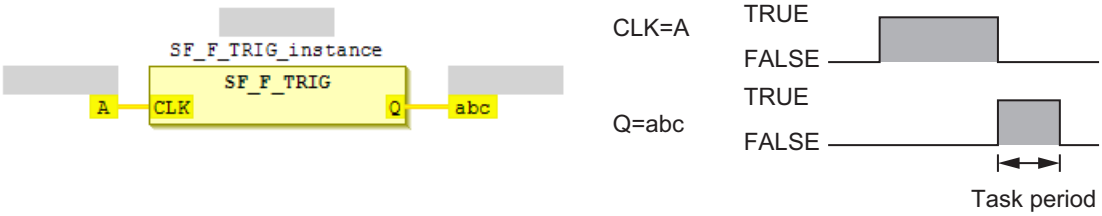
Output Variables

Variable	Name	Data type	Valid range	Default	Description
Q	Output signal	SAFEBOOL	TRUE or FALSE	FALSE	Output signal

Function

- F_TRIG assigns TRUE to output signal *Q* for one task period only when input signal *CLK* changes to FALSE. Otherwise, the value of *Q* is FALSE.
- If the value of *CLK* is FALSE when the power supply is turned ON, the value of *Q* changes to TRUE.

The following figure shows a programming example and timing chart.

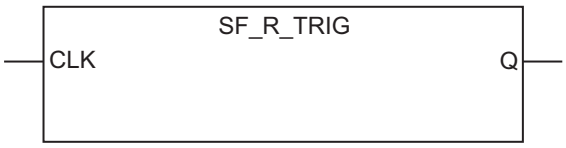


Additional Information

The SF_F_TRIG instruction assigns TRUE to output signal *Q* for only one task period when the power supply is turned ON (RUN).

SF_R_TRIG

This FB outputs TRUE for one task period only when the input signal changes to TRUE.

Instruction	Name	FB/FUN	Graphic expression
SF_R_TRIG	Up Trigger	FB	

Variables

Input Variables

Variable	Name	Data type	Valid range	Default	Description
CLK	Input signal	SAFEBOOL	TRUE or FALSE	FALSE	Input signal

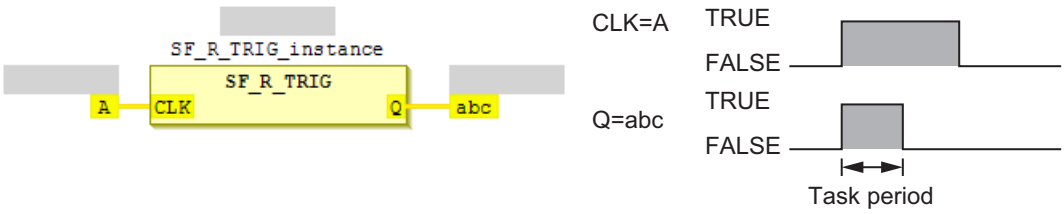
Output Variables

Variable	Name	Data type	Valid range	Default	Description
Q	Output signal	SAFEBOOL	TRUE or FALSE	FALSE	Output signal

Function

- This FB assigns TRUE to output signal *Q* for one task period only when input signal *CLK* changes to TRUE. Otherwise, the value of *Q* is FALSE.
- If the value of *CLK* is TRUE when the power supply is turned ON, the value of *Q* changes to TRUE.

The following figure shows a programming example and timing chart.



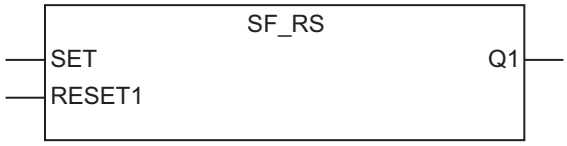
Additional Information

The SF_R_TRIG instruction assigns TRUE to output signal *Q* for only one task period when the power supply is turned ON (RUN).

SF_RS

This FB retains the value of a SAFEBOOL variable.

It gives priority to the *Reset* input if both the *Set* input and *Reset* input are TRUE.

Instruction	Name	FB/FUN	Graphic expression
SF_RS	Reset-Priority Keep	FB	

Variables

Input Variables

Variable	Name	Data type	Valid range	Default	Description
SET	Set	SAFEBOOL	TRUE or FALSE	FALSE	Set input
RESET1	Reset	BOOL	TRUE or FALSE	FALSE	Reset input (Given priority.)

Output Variables

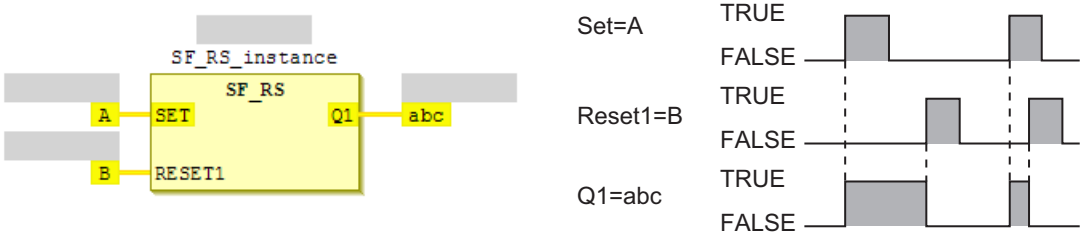
Variable	Name	Data type	Valid range	Default	Description
Q1	Keep	SAFEBOOL	TRUE or FALSE	FALSE	Keep output

Function

- This instruction forms a self-holding output that gives priority to resetting. The following table shows the relationship between the inputs and outputs.

Value of *SET*	Value of *RESET1*	Value of *Q1*
TRUE	TRUE	FALSE
TRUE	FALSE	TRUE
FALSE	TRUE	FALSE
FALSE	FALSE	Not changed.

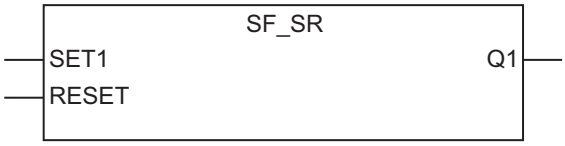
The following figure shows a programming example and timing chart.



SF_SR

This FB retains the value of a SAFEBOOL variable.

It gives priority to the *Set* input if both the *Set* input and *Reset* input are TRUE.

Instruction	Name	FB/FUN	Graphic expression
SF_SR	Set-Priority Keep	FB	

Variables

Input Variables

Variable	Name	Data type	Valid range	Default	Description
SET1	Set	SAFEBOOL	TRUE or FALSE	FALSE	Set input (Given priority.)
RESET	Reset	BOOL	TRUE or FALSE	FALSE	Reset input

Output Variables

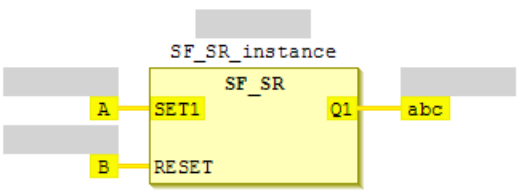
Variable	Name	Data type	Valid range	Default	Description
Q1	Keep	SAFEBOOL	TRUE or FALSE	FALSE	Keep output

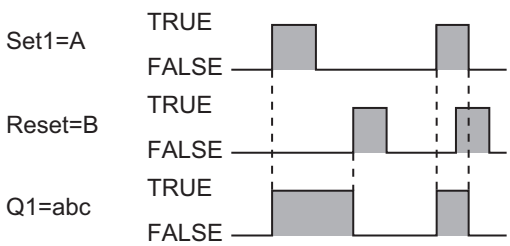
Function

- This instruction forms a self-holding output that gives priority to setting. The following table shows the relationship between the inputs and outputs.

Value of *SET1*	Value of *RESET*	Value of *Q1*
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	FALSE
FALSE	FALSE	Not changed.

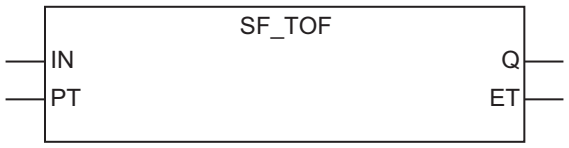
The following figure shows a programming example and timing chart.





SF_TOF

This FB outputs FALSE when the set time elapses after the timer starts.

Instruction	Name	FB/FUN	Graphic expression
SF_TOF	Off-Delay Timer	FB	

Variables

Input Variables

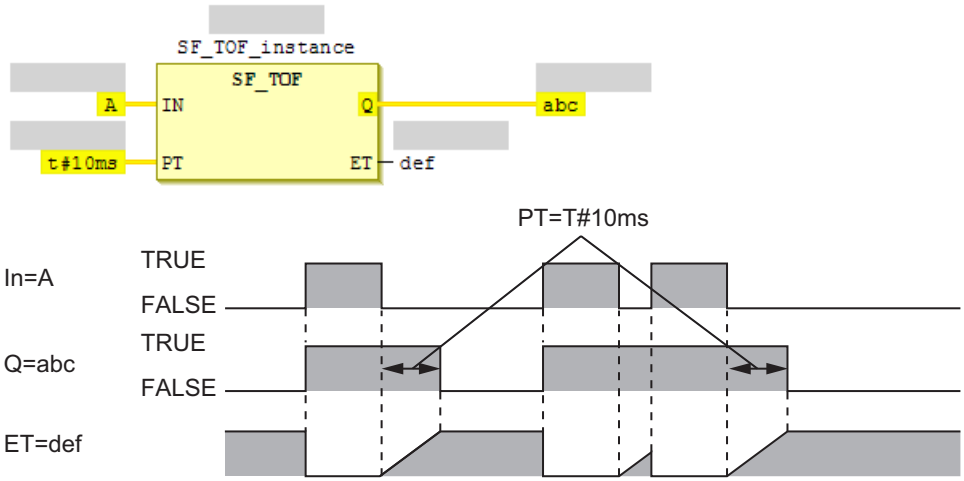
Variable	Name	Data type	Valid range	Default	Description
IN	Timer input	BOOL	TRUE or FALSE	FALSE	TRUE: Timer reset specification FALSE: Timer start specification
PT	Set time	TIME	Depends on data type.	0	Time from when timer starts until *Q* changes to FALSE

Output Variables

Variable	Name	Data type	Valid range	Default	Description
Q	Timer output	SAFEBOOL	TRUE or FALSE	FALSE	TRUE: *IN* is TRUE and *ET* is lower than *PT* after the timer starts. FALSE: *ET* reached *PT*.
ET	Elapsed time	TIME	Depends on data type.	0	Elapsed time since timer started

Function

- This FB outputs FALSE when the set time elapses after the timer starts. The time is set in milliseconds.
- The timer starts when timer input *IN* changes to FALSE. Elapsed time *ET* is incremented as time elapses.
- When *ET* reaches set time *PT*, timer output *Q* changes to FALSE. *ET* is not incremented after that.
- The timer is reset when *IN* changes to TRUE. *ET* changes to 0 and *Q* changes to TRUE.
  If the timer is started and then *IN* changes to FALSE before *ET* reaches *PT*, the timer is reset.
- The following figure shows a programming example and timing chart for a *PT* of T#10ms.
  Variable *abc* will change to FALSE 10 ms after variable *A* changes to FALSE.

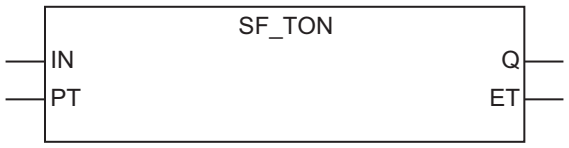


Additional Information

- Use the SF_TP instruction for a timer that changes the timer output to TRUE when timing starts and then changes the timer output to FALSE when the set time is reached.
- Use the SF_TON instruction for a timer that starts when *IN* changes to TRUE and then changes the timer output to TRUE when the elapsed time reaches the set time.

SF_TON

This FB outputs TRUE when the set time elapses after the timer starts.

Instruction	Name	FB/FUN	Graphic expression
SF_TON	On-Delay Timer	FB	

Variables

Input Variables

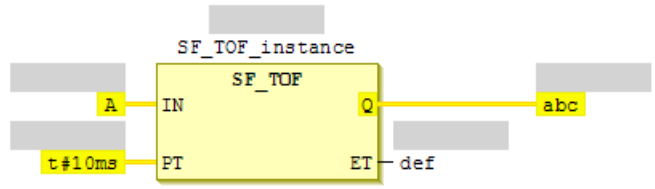
Variable	Name	Data type	Valid range	Default	Description
IN	Timer input	BOOL	TRUE or FALSE	FALSE	TRUE: Timer start specification FALSE: Timer reset specification
PT	Set time	TIME	Depends on data type.	0	Time from when timer starts until *Q* changes to TRUE

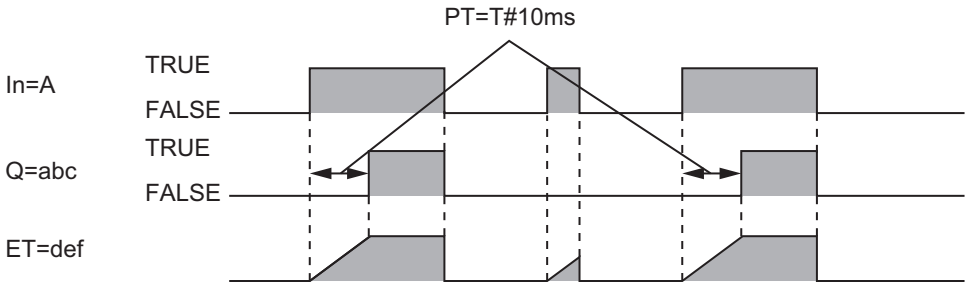
Output Variables

Variable	Name	Data type	Valid range	Default	Description
Q	Timer output	SAFEBOOL	TRUE or FALSE	FALSE	TRUE: *ET* reached *PT*. FALSE: *IN* is TRUE and *ET* is lower than *PT* after the timer starts.
ET	Elapsed time	TIME	Depends on data type.	0	Elapsed time since timer started

Function

- This FB outputs TRUE when the set time elapses after the timer starts. The time is set in milliseconds.
- The timer starts when timer input *IN* changes to TRUE. Elapsed time *ET* is incremented as time elapses.
- When *ET* reaches set time *PT*, timer output *Q* changes to TRUE. *ET* is not incremented after that.
- The timer is reset when *IN* changes to FALSE. *ET* changes to 0 and *Q* changes to FALSE.
  If the timer is started and then *IN* changes to FALSE before *ET* reaches *PT*, the timer is reset.
- The following figure shows a programming example and timing chart when *PT* is T#10ms. Variable *abc* will change to TRUE 10 ms after variable *A* changes to TRUE.





Additional Information

- Use the SF_TP instruction for a timer that changes the timer output to TRUE when timing starts and then changes the timer output to FALSE when the set time is reached.
- Use the SF_TOF instruction for a timer that starts when *IN* changes to FALSE and then changes the timer output to FALSE when the elapsed time reaches the set time.

SF_TP

This FB outputs TRUE during the set time after the timer starts.

Instruction	Name	FB/FUN	Graphic expression
SF_TP	Timer Pulse	FB	

Variables

Input Variables

Variable	Name	Data type	Valid range	Default	Description
IN	Timer input	BOOL	TRUE or FALSE	FALSE	TRUE: Timer start specification FALSE: Timer reset specification
PT	Set time	TIME	Depends on data type.	0	Time that *Q* remains at TRUE

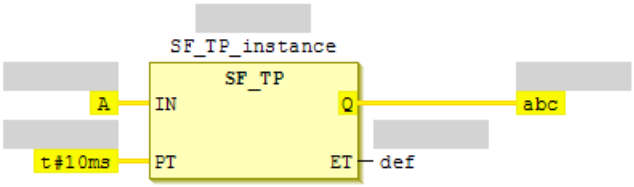
Output Variables

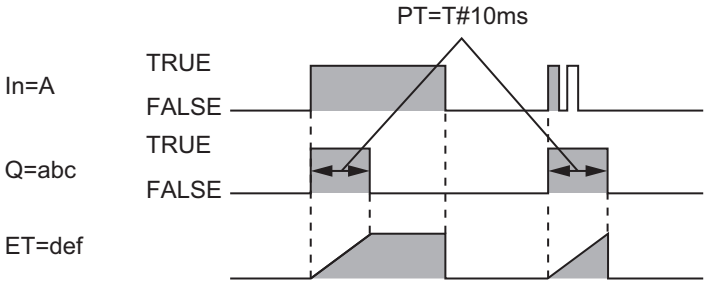
Variable	Name	Data type	Valid range	Default	Description
Q	Timer output	SAFEBOOL	TRUE or FALSE	FALSE	TRUE: *IN* is TRUE and *ET* is lower than *PT* after the timer starts. FALSE: *ET* reached *PT*.
ET	Elapsed time	TIME	Depends on data type.	0	Elapsed time since timer started

Function

- This FB outputs TRUE during the set time after the timer starts. The time is set in milliseconds.
- The timer starts when timer input *IN* changes to TRUE and timer output *Q* changes to TRUE. Elapsed time *ET* is incremented as time elapses.
- When *ET* reaches set time *PT*, timer output *Q* changes to FALSE. *ET* is not incremented after that.
- The timer is reset when *IN* changes to FALSE. *ET* changes to 0.
  The timer is not reset even if *IN* changes to FALSE after the timer starts but before *ET* reaches *PT*.
- The following figure shows a programming example and timing chart for a *PT* of T#10ms.

  Variable *abc* changes to TRUE as soon as variable *A* changes to TRUE. Variable *abc* changes to FALSE 10 ms later.





Additional Information

- Use the SF_TON instruction for a timer that starts when *IN* changes to TRUE and then changes the timer output to TRUE when the elapsed time reaches the set time.
- Use the SF_TOF instruction for a timer that starts when *IN* changes to FALSE and then changes the timer output to FALSE when the elapsed time reaches the set time.

4

Safety Function Blocks

This section gives the specifications of the safety function blocks that you can use for NX-series Safety Control Units.

General Rules for Safety Function Blocks

This section gives the general rules for safety function blocks. Safety function block is abbreviated as “safety FB” and “function block” is abbreviated as “FB.”

Rules That Are Specific to Safety FBs

Item	Rule
Default signal	The default for all SAFEBOOL signals is FALSE (i.e., the safe state).
Signal levels	The values of SAFEBOOL variables have the following meanings. • FALSE: Indicates the safe state for a system output. • TRUE: Indicates that operation is correct in terms of system safety (e.g., that correct operation is possible). These definitions reflect the functionality in an IEC 61131 standard. For example, if an error occurs, all outputs change to FALSE as defined in the default signal rule.
Outputs	All outputs are refreshed each safety task period.
Omitting I/O parameters	You can omit parameters. Defaults are applied for any omitted parameters. However, these defaults will not lead to a non-safe state for the system under any circumstances. Defaults are defined for attributes (i.e., for variables or constants) and for FBs.
Start processing	Outputs are initially set to the default values. Outputs are enabled after the first FB call.
Error handling and diagnosis	All safety FBs have two error-related outputs: *Error* and *DiagCode*.

Safety FB Common Input Variables

The common input variables for safety FBs are listed in the following table.

Input parameter name	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Enables and disables the FB. You can input a variable or a constant. The default is FALSE. You can input a variable that gives the status of the input device to evaluate for this parameter in order to disable the FB and to prevent unrelated diagnostic data from being output. If this parameter is FALSE, all output variables take their default values. Normally, input a TRUE constant to enable the FB.

Input parameter name	Data type	Valid range	Default	Description
S_StartReset	SAFEBOOL	TRUE or FALSE	FALSE	Controls automatic and manual resetting at startup (i.e., when program execution is started). You can input a variable or a constant. FALSE (default): Perform resetting manually when the Safety CPU Unit is started. TRUE: Resetting is performed automatically when the Safety CPU Unit is started. Use automatic resetting only when you can verify that no hazard of any sort will result from automatically resetting the Safety CPU Unit.
S_AutoReset	SAFEBOOL	TRUE or FALSE	FALSE	Controls automatic and manual resetting for the operation of an emergency stop button. You can input a variable or a constant. FALSE (default): Perform resetting manually when the emergency stop button is released. TRUE: Resetting is performed automatically when the emergency stop button is released. Use automatic resetting only when you can verify that no hazard of any sort will result from automatically resetting the Safety CPU Unit.
Reset	BOOL	TRUE or FALSE	FALSE	The reset input. Input a variable. This parameter is used for different purposes for different FBs. • You can use it to reset the function block status and release the relationship between the error and code that is returned in *DiagCode* after the cause of the error is removed. • You can use it for a manual reset with an operator restart interlock. The reset processing must be designed to reset the FB. • You can use it for other FB-specific resets. Resetting is effective only when the signal changes from FALSE to TRUE. Resetting is not performed unless the change from FALSE to TRUE is detected. Some FBs detect this as an instruction execution error. This is described in *Instruction Execution Errors* under *Function* for the relevant FB. You must connect a SAFEBOOL variable (not a BOOL variable) depending on safety requirements.

Safety FB Common Output Variables

The common output variables for safety FBs are listed in the following table.

Output variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	The ready flag. FALSE: Indicates that the FB is not active and the program is not executed. This is useful in DEBUG Mode or to activate/deactivate additional FBs, as well as for further processing in the functional program. TRUE: Indicates that the FB is operating and that the output results have been stored. This variable is used for debugging or for further processing in the user program.
Error	BOOL	TRUE or FALSE	FALSE	The error flag. FALSE: Indicates that no error has occurred. The state is given by *DiagCode*. TRUE: Indicates that an error has occurred. The error state is given by *DiagCode*.
DiagCode	WORD	Depends on FB state code.	16#0000	Diagnostic information. All status (active, non-active, and error) for the FB is stored in this variable. The information is given as a hexadecimal number. Only one code is given each time. If more than one error has occurred, the information for the first error that is detected is output to *DiagCode*. Refer to *Diagnostic Codes* on page 4-4, below, for details. This variable is used for debugging or for further processing in the user program.

Diagnostic Codes

All FBs output unique diagnostic information to *DiagCode*.
If an error does not occur, *DiagCode* gives the internal state of the FB. Errors are given in hexadecimal. You can get detailed information on internal and external errors for FBs from *DiagCode*. More than one reset input may be required to reset a FB.

Range of Values for Safety FB Common Diagnostic Codes

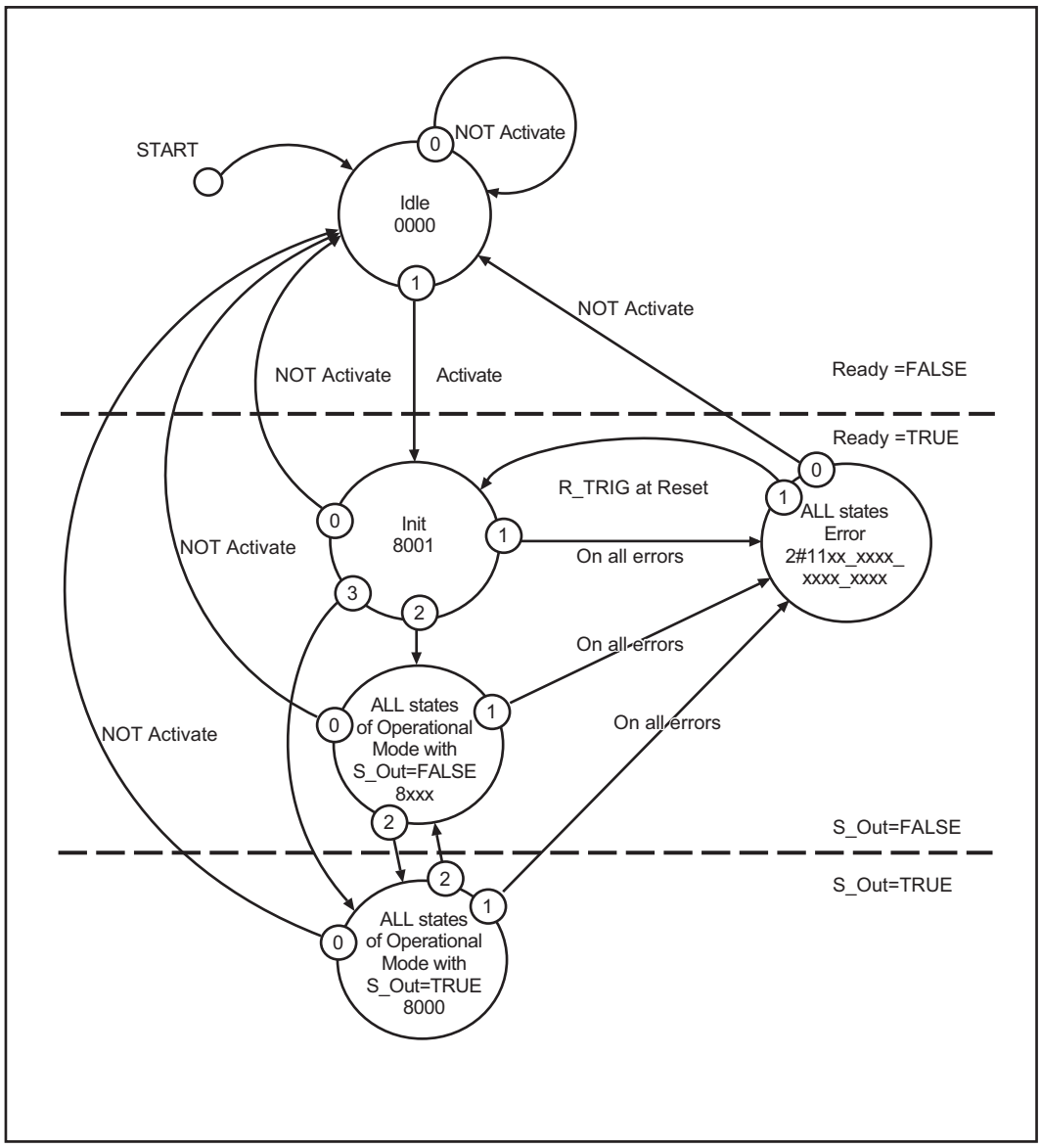
DiagCode	Meaning
0000_0000_0000_0000 binary	The FB is not operating or the Safety CPU Unit is stopped.
10xx_xxxx_xxxx_xxxx binary	The FB is operating and an error does not exist. “x” is a FB-specific code.
11xx_xxxx_xxxx_xxxx binary	The FB is operating and an error exists. “x” is FB-specific code.

Safety FB Common Diagnostic Codes

DiagCode	Meaning
0000_0000_0000_0000 binary 0000 hex 0 decimal	Indicates the Idle state. In this state, the FB is not operating. Normally, the I/O have the following status. • *Activate* = FALSE • Safety input variables: TRUE or FALSE • *Ready* = FALSE • *Error* = FALSE • Safety output variables: FALSE
1000_0000_0000_0000 binary 8000 hex 32,768 decimal	Indicates the default operating status for normal operation that makes the safety output variable TRUE. The FB is operating and no error has occurred, or the operation is in a different state that makes the safety output FALSE. Normally, the I/O have the following status. • *Activate* = TRUE • Safety input variables: TRUE • *Ready* = TRUE • *Error* = FALSE • Safety output variables: TRUE
1000_0000_0000_0001 binary 8001 hex 32,769 decimal	Indicates the Init state. The FB detected *Activate*, but the safety output is set to FALSE. Normally, the I/O have the following status. • *Activate* = TRUE • Safety input variables: TRUE or FALSE • *Ready* = TRUE • *Error* = FALSE • Safety output variables: FALSE
1000_0000_0000_0010 binary 8002 hex 32,770 decimal	The FB is operating and a safety request input was detected. For example a safety input variable is FALSE. The safety output variable is FALSE. After this state, a safety input request is received and the safety output is disabled. Normally, the I/O have the following status. • *Activate* = TRUE • Safety input variables: FALSE • *Ready* = TRUE • *Error* = FALSE • Safety output variables: FALSE
1000_0000_0000_0011 binary 8003 hex 32,771 decimal	The safety output from an operating FB was disabled by a safety request and the safety output is FALSE. The safety request has been canceled, but the safety output remains FALSE until the reset input is detected. Normally, the I/O have the following status. • *Activate* = TRUE • Safety input variables: Change from FALSE to TRUE (and remain TRUE) • *Ready* = TRUE • *Error* = FALSE • Safety output variables: FALSE

Safety FB Common State Transition Diagram

The following type of transition diagram shows changes in the state of the safety FB. This section describes how to interpret state transition diagrams.



- The above diagram outlines the state transitions that apply to all safety FBs. Transitions that have specific meanings for some FBs are not given here. They are described individually for the applicable FBs.
- This diagram is separated into three parts.
  In the top part, the FB is not operating and is in the safe state (i.e., safety outputs are FALSE).
  In the middle part, the FB is operating and is in the safe state (i.e., safety outputs are FALSE).
  In the bottom part, the FB is operating normally (i.e., safety outputs are TRUE).
- The dotted line at the top of the state transition diagram indicates transitions from not active to active. The dotted line at the bottom of the diagram indicates transitions from the safe state to the normal state of the FB.
- The priority of parallel transitions are shown with numbers. The highest priority is 0.
- The circles that indicate the states give the status name and the hexadecimal value of *DiagCode*.
- OR, AND, and XOR are used as logical operators and NOT is used as the logical negator to indicate status.

- In FB descriptions, the startup state is the Idle state. This state changes to an operating state only after entering the Init state.
- You can change *Activate* to FALSE to enter the Idle state from any other state.
  If *Activate* is FALSE, operation 0 has the highest priority. These transitions are not shown on the state transition diagrams. They are described in the footnotes for individual state transition diagrams.
- The output results are not given in the state transition diagrams. The status description and output results of a FB are given in *FB-specific Error Codes* and *FB-specific State Codes (No Error)* on page 4-7.

● FB-specific Error Codes

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
Cxxx	49152 to 53247	Error	*Ready* = TRUE Safety output variables: FALSE *Error* = TRUE

● FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	*Ready* = FALSE Safety output variables: FALSE *Error* = FALSE
8001	32769	Init	*Ready* = TRUE Safety output variables: FALSE Error = FALSE
8xxx	32769 to 36863	All states of operation mode where safety output variable is FALSE	*Ready* = TRUE Safety output variable = FALSE *Error* = FALSE
8000	32768	All states of operation mode where safety output variable is TRUE	*Ready* = TRUE Safety output variables: TRUE *Error* = FALSE

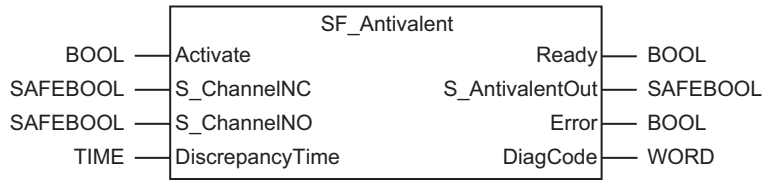
Safety Function Block Instructions

Instruction	Name	Function	Page
SF_Antivalent	Antivalent	Monitors the discrepancy time for two antivalent SAFEBOOL inputs.	P. 4-9
SF_EDM	External Device Monitoring	Controls a safety output and monitors actuator control.	P. 4-15
SF_EmergencyStop	Emergency Stop	Monitors the input from an emergency stop button.	P. 4-23
SF_EnableSwitch	Enable Switch	Supports stopping a safety protection function that uses an enable switch.	P. 4-30
SF_Equivalent	Equivalent	Monitors the discrepancy time for two equivalent SAFEBOOL inputs.	P. 4-36
SF_ESPE	Electro-Sensitive Protective Equipment (ESPE)	Monitors electro-sensitive protective equipment (ESPE).	P. 4-42
SF_GuardLocking	Safety Guard Interlocking with Locking	Controls entry to a hazardous area with a four-state interlock guard with a guard lock.	P. 4-49
SF_GuardMonitoring	Safety Guard Monitoring	Monitors a relevant safety guard and opens/closes the safety guard.	P. 4-55
SF_ModeSelector	Mode Selector	Selects the system operation mode (automatic, manual, semi-automatic, etc.).	P. 4-61
SF_MutingPar	Parallel Muting	Performs parallel muting with four muting sensors.	P. 4-70
SF_MutingPar_2Sensor	Parallel Muting with 2 Sensors	Performs parallel muting with two muting sensors.	P. 4-82
SF_MutingSeq	Sequential Muting	Performs sequential muting with four muting sensors.	P. 4-91
SF_OutControl	Out Control	Controls a safety output with a control signal and safety signal from a function application.	P. 4-101
SF_SafetyRequest	Safety Request	Makes requests for the safe state and monitors the safe state for an actuator (e.g., a drive or valve) that has a safety function.	P. 4-107
SF_TestableSafetySensor	Testable Safety Sensors	Tests functionality with the external test function of electro-sensitive protective equipment (ESPE).	P. 4-113
SF_ TwoHandControlTypeII	Two-Hand Control Type II	Provides a type II, two-hand control function as defined in ISO 13851(EN 574).	P. 4-124
SF_ TwoHandControlTypeIII	Two-Hand Control Type III	Provides a type III, two-hand control function as defined in ISO 13851(EN 574).	P. 4-129

SF_Antivalent

This safety FB monitors the discrepancy time for two antivalent SAFEBOOL inputs.

Note "Antivalent" refers to the state where two inputs are simultaneously in the opposite status during normal operation. This kind of inputs is also called complementary or non-equivalent.

Instruction	Name	FB/FUN	Graphic expression
SF_Antivalent	Antivalent	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_ChannelNC	SAFEBOOL	TRUE or FALSE	FALSE	It functions as the input for a N.C. connection. FALSE: The N.C. contacts are open. TRUE: The N.C. contacts are closed. * N.C. = Normally closed
S_ChannelNO	SAFEBOOL	TRUE or FALSE	TRUE	It functions as the input for a N.O. connection. FALSE: The N.O. contacts are open. TRUE: The N.O. contacts are closed. * N.O. = Normally open
Discrepancy-Time	TIME	Depends on data type.	T#0ms	It sets the maximum monitoring time for discrepancy between two inputs.

Output Variables

Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_AntivalentOut	SAFEBOOL	TRUE or FALSE	FALSE	The safety output. FALSE: At least one of the signals is FALSE or the state changed outside the monitoring time. TRUE: The two input signals are active and the state changed within the monitoring time.
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

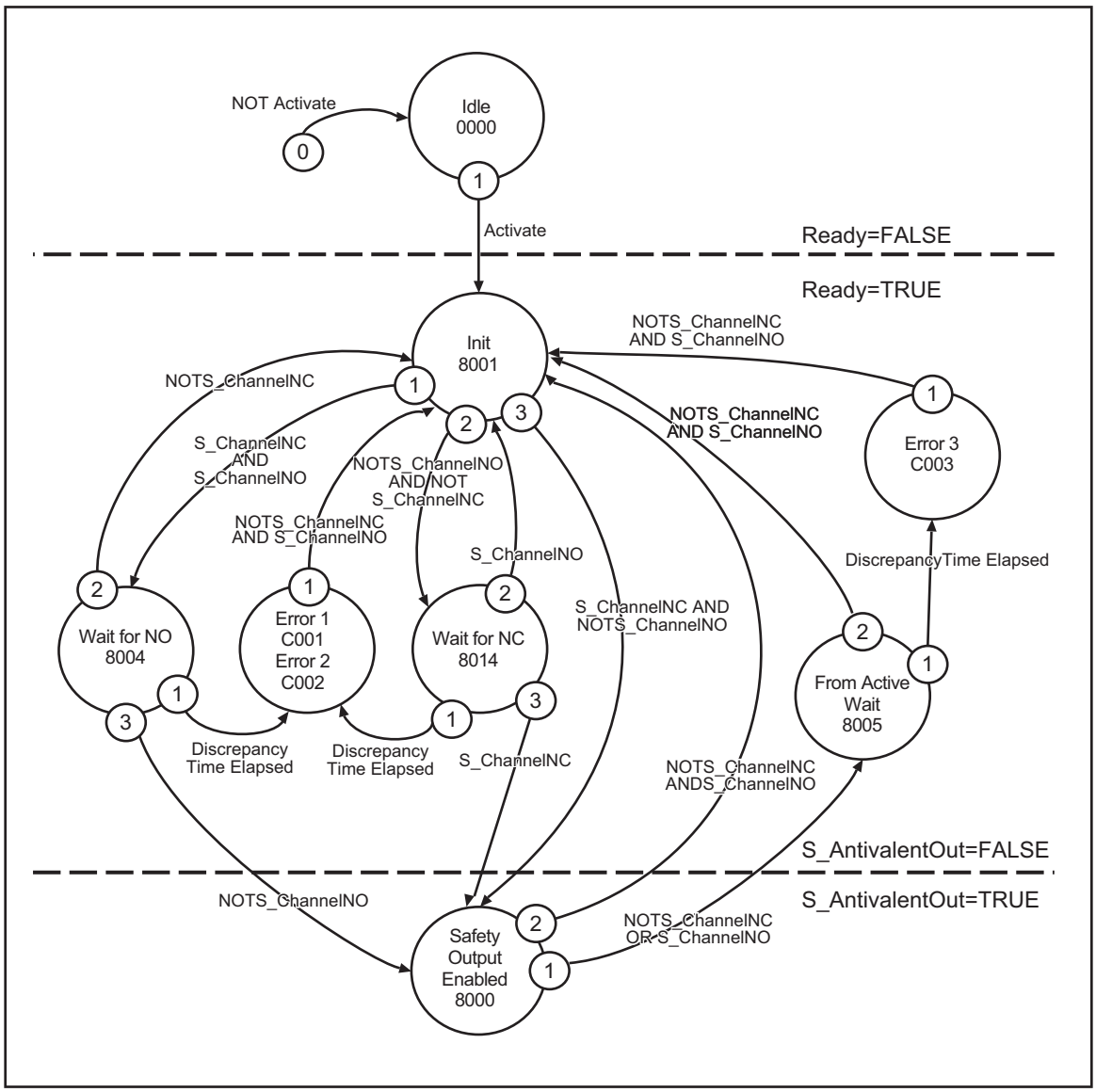
Function

- This FB monitors the time that two SAFEBOOL inputs are the same and outputs the result on the SAFEBOOL output when they are different.
- *S_ChannelNC* and *S_ChannelNO* are dependent on each other. The evaluation result for both channels is output.
- When the input for one channel changes so that the two channel inputs are no longer different, the FB starts monitoring the discrepancy time. An error occurs if a change does not occur that makes them different again within the monitoring time.
- If *S_AntivalentOut* is TRUE and the signal for one channel changes, the output immediately changes to FALSE.
- If an error occurs, make the inputs for both channels inactive (make *S_ChannelNC* FALSE and make *S_ChannelNO* TRUE) to reset the FB.
- Set *DiscrepancyTime* to a value that is longer than the safety task period.
  Refer to the *NX-series Safety Control Unit User's Manual* (Cat. No. Z930) for application methods for *DiscrepancyTime*.

Precautions for Correct Use

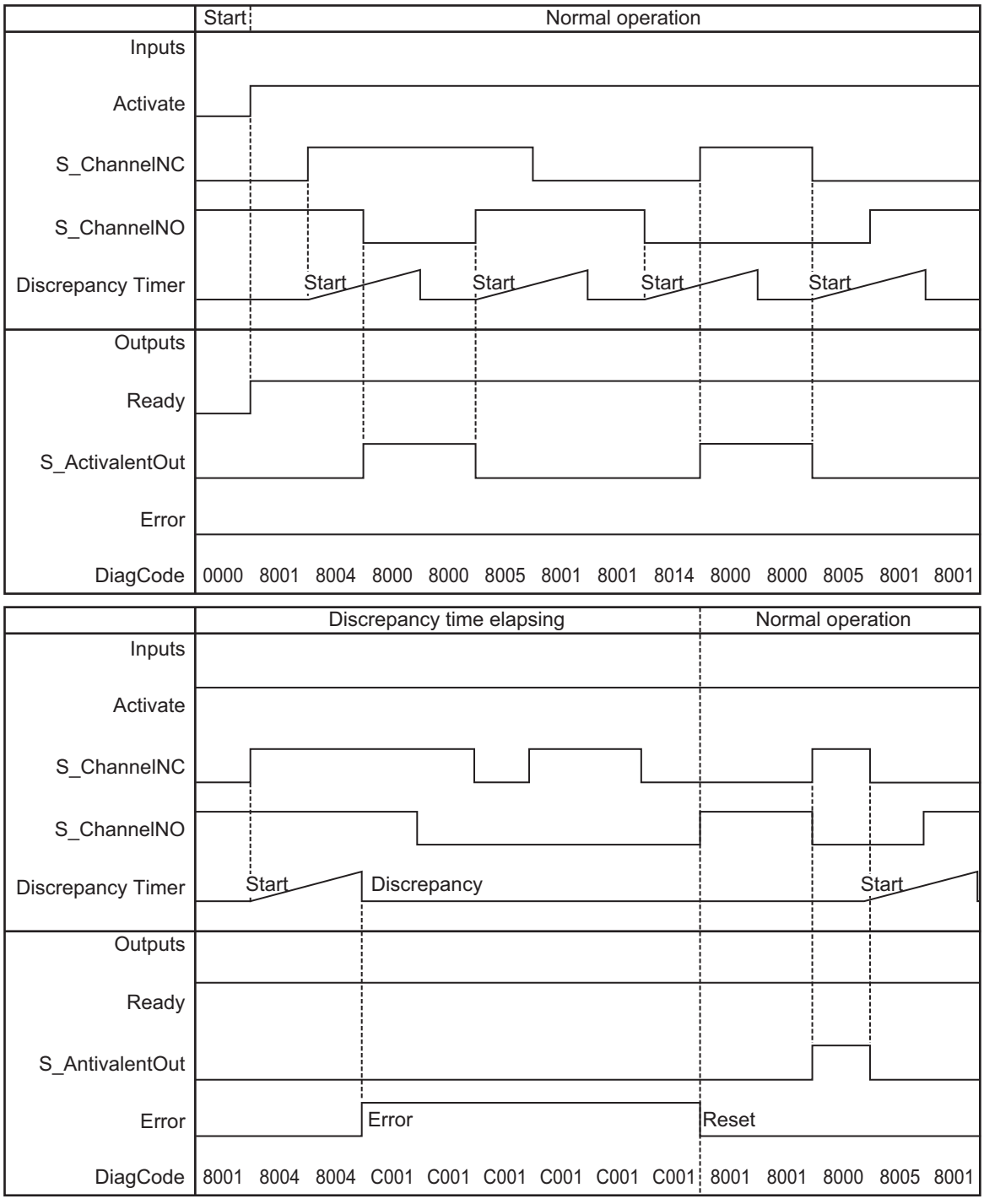
This FB does not have a restart interlock. You must connect it to a FB that has a restart interlock.

State Transition Diagram



Note Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

Timing Charts



Instruction Execution Errors

Error Detected

The FB monitors the discrepancy time between *S_ChannelNC* and *S_ChannelNO*.

Operation for Errors

- If an error is detected, *S_AntivalentOut* changes to FALSE and *Error* changes to TRUE. *Diag-Code* shows the error state.
- If an error occurs in an input, make the inputs for both channels inactive (make *S_ChannelNC* FALSE and make *S_ChannelNO* TRUE) to reset the FB.

FB-specific Error Codes

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C001	49153	Error 1	An input did not occur within the monitoring time in the Wait for NO state (8004). *Ready* = TRUE *S_AntivalentOut* = FALSE *Error* = TRUE
C002	49154	Error 2	An input did not occur within the monitoring time in the Wait for NC state (8014). *Ready* = TRUE *S_AntivalentOut* = FALSE *Error* = TRUE
C003	49155	Error 3	The input did not change within the monitoring time while the state changes from the From Active Wait (8005) to the Init (8001) state. *S_ChannelNO* did not change to TRUE after *S_ChannelNC* changed to FALSE. Or, *S_ChannelNC* did not change to FALSE after *S_ChannelNO* changed to TRUE. *Ready* = TRUE *S_AntivalentOut* = FALSE *Error* = TRUE

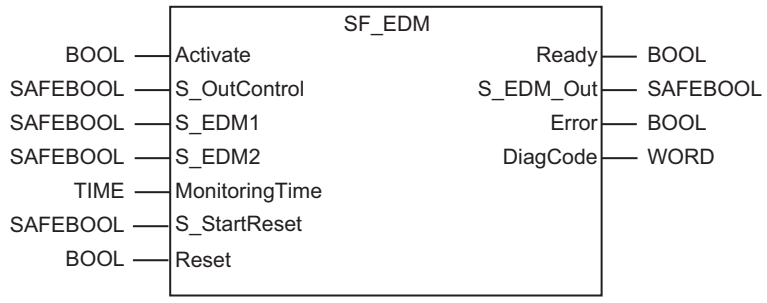
FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *S_AntivalentOut* = FALSE *Error* = FALSE
8001	32769	Init	The FB detected an activate signal and the FB is active. *Ready* = TRUE *S_AntivalentOut* = FALSE *Error* = FALSE
8000	32768	Safety Output Enabled	An input changed to TRUE in Antivalent Mode. *Ready* = TRUE *S_AntivalentOut* = TRUE *Error* = FALSE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8004	32772	Wait for NO	*S_ChannelNC* changed to TRUE, the discrepancy time timer started operation, and the FB is waiting for *S_ChannelNO* to change to FALSE. *Ready* = TRUE *S_AntivalentOut* = FALSE *Error* = FALSE
8014	32788	Wait for NC	*S_ChannelNO* changed to FALSE, the discrepancy time timer started operation, and the FB is waiting for *S_ChannelNC* to change to TRUE. *Ready* = TRUE *S_AntivalentOut* = FALSE *Error* = FALSE
8005	32773	From Active Wait	One of the channels changed to FALSE, the discrepancy time timer started operation, and the FB is waiting for the other channel to change to FALSE. *Ready* = TRUE *S_AntivalentOut* = FALSE *Error* = FALSE

SF_EDM

This safety FB controls a safety output and monitors actuator control.

Instruction	Name	FB/FUN	Graphic expression
SF_EDM	External Device Monitoring	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_OutControl	SAFEBOOL	TRUE or FALSE	FALSE	A variable. The control signal from the previous safety FB. The signal from a typical FB from the library (SF_OutControl, SF_TwoHandControlTypeII, etc.) is used. FALSE: Disables the safety output (S_EDM_Out). TRUE: Enables the safety output (S_EDM_Out).
S_EDM1	SAFEBOOL	TRUE or FALSE	FALSE	A variable. The feedback signal from the first connected actuator. FALSE: The first connected actuator is in switched status. TRUE: The first connected actuator is in its default status.
S_EDM2	SAFEBOOL	TRUE or FALSE	FALSE	A variable. The feedback signal from the second connected actuator. If there is only one feedback signal used in the application, connect it to both *S_EDM1* and *S_EDM2*. FALSE: The second connected actuator is in switched status. TRUE: The second connected actuator is in its default status.
MonitoringTime	TIME	Depends on data type.	T#0ms	A constant. It gives the maximum response time for the connected, monitored actuator.
S_StartReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
Reset	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.

Output Variables

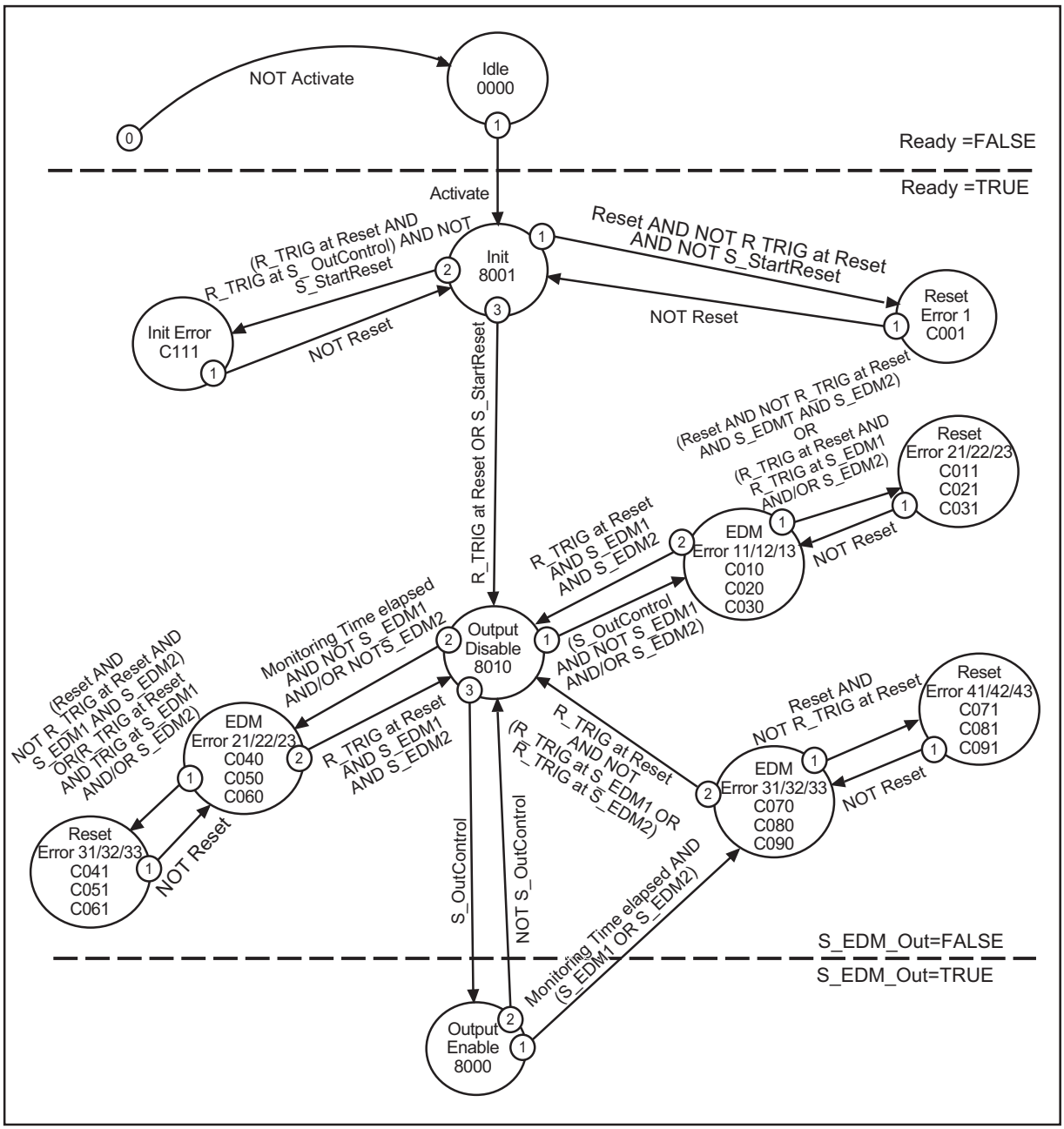
Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_EDM_Out	SAFEBOOL	TRUE or FALSE	FALSE	Controls the actuator. It monitors the result with the feedback signal *S_EDMx*. FALSE: Disables the connected actuator. TRUE: Enables the connected actuator.
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

Function

Introduction

- This FB controls a safety output and monitors actuator control.
- This FB monitors the initial status of the actuator through feedback signals (*S_EDM1* and *S_EDM2*) before the actuator is activated by the FB.
- After the actuator is activated by this FB, the FB also monitors the actuator's switched status (*MonitoringTime*).
- Two single feedback signals must be used for an exact diagnosis of the connected actuators. A common feedback signal from the two connected actuators must be used for a restricted yet simple diagnostic function of the connected actuators. To achieve that, you must connect the common signal to both the *S_EDM1* and *S_EDM2* parameters. Therefore, *S_EDM1* and *S_EDM2* will be controlled by the same signal.
- The switching device for which the safety function is used must be selected from the category that was determined by risk assessment.
- Activate the *S_StartReset* input only when you can ensure that no hazardous state will occur as the result of starting the Safety CPU Unit.

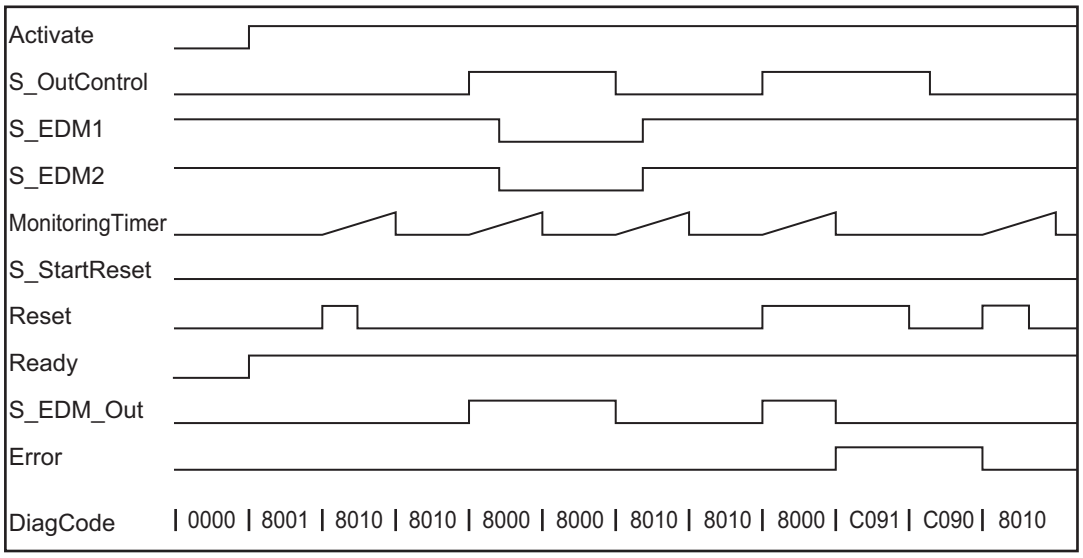
State Transition Diagram



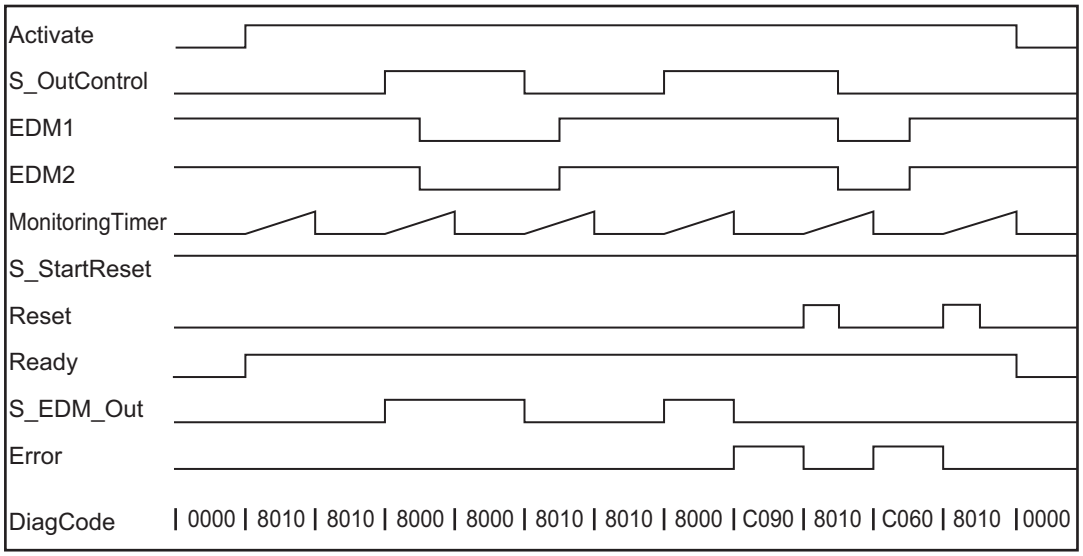
Note Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

Timing Charts

S_StartReset = FALSE



S_StartReset = TRUE



Instruction Execution Errors

Error Detected

The following conditions force a transition to an error state.

- An invalid process always-TRUE *Reset* signal
- An invalid process EDM signal
- Programming error that results in incorrect interconnections between *S_OutControl* and *Reset*

Operation for Errors

- If an error occurs, *S_EDM_Out* changes to FALSE and the safe state is maintained. *Error* changes to TRUE and *DiagCode* shows the error state.
- You must change *Reset* to TRUE to reset EDM error messages.
- You can change *Reset* to FALSE to reset error messages for *Reset.*
  After the FB is started, you can change the *Reset* input to TRUE to reset the optional startup inhibit.

FB-specific Error Codes

DiagCode (hexa-decimal)	DiagCode (decimal)	Status name	Status description and output results
C001	49153	Reset Error 1	When the Init state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C011	49169	Reset Error 21	An undetected change to TRUE was detected for *EDM1* and *Reset* or equivalent signals were detected during EDM Error 11 status. (Both *Reset* and *EDM1* changed to TRUE at the same time.) *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C021	49185	Reset Error 22	An undetected change to TRUE was detected for *EDM2* and *Reset* or equivalent signals were detected during EDM Error 12 status. (Both *Reset* and *EDM2* changed to TRUE at the same time.) *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C031	49201	Reset Error 23	An undetected change to TRUE was detected for *EDM1, EDM2,* and *Reset* or equivalent signals were detected during EDM Error 13 status. (*Reset, EDM1,* and *EDM2* changed to TRUE at the same time.) *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE

DiagCode (hexa-decimal)	DiagCode (decimal)	Status name	Status description and output results
C041	49217	Reset Error 31	An undetected change to TRUE was detected for *EDM1* and *Reset* or equivalent signals were detected during EDM Error 21 status. (Both *Reset* and *EDM1* changed to TRUE at the same time.) *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C051	49233	Reset Error 32	An undetected change to TRUE was detected for *EDM2* and *Reset* or equivalent signals were detected during EDM Error 22 status. (Both *Reset* and *EDM2* changed to TRUE at the same time.) *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C061	49249	Reset Error 33	An undetected change to TRUE was detected for *EDM1, EDM2,* and *Reset* or equivalent signals were detected during EDM Error 23 status. (*Reset, EDM1,* and *EDM2* changed to TRUE at the same time.) *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C071	49265	Reset Error 41	When EDM Error 31 status was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C081	49281	Reset Error 42	When EDM Error 32 status was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C091	49297	Reset Error 43	When EDM Error 33 status was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C010	49168	EDM Error 11	The *EDM1* signal is not valid during the initial status of the actuator. If *S_OutControl* is enabled when output is not possible, the *EDM1* signal changes to FALSE. *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C020	49184	EDM Error 12	The *EDM2* signal is not valid during the initial status of the actuator. If *S_OutControl* is enabled when output is not possible, the *EDM2* signal changes to FALSE. *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE

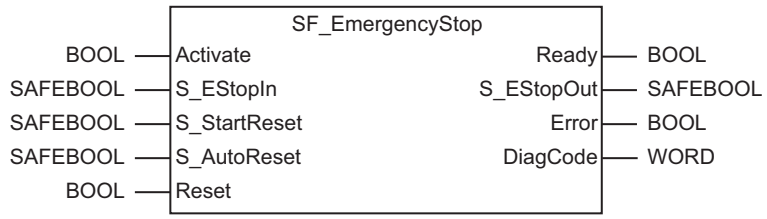
DiagCode (hexa-decimal)	DiagCode (decimal)	Status name	Status description and output results
C030	49200	EDM Error 13	The *EDM1* and *EDM2* signals are not valid during the initial status of the actuator. If *S_OutControl* is enabled when output is not possible, the *EDM1* and *EDM2* signals change to FALSE. *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C040	49216	EDM Error 21	The *EDM1* signal is not valid during the initial status of the actuator. The *EDM1* signal changed to FALSE when output was not possible, and the monitoring time ended. *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C050	49232	EDM Error 22	The *EDM2* signal is not valid during the initial status of the actuator. The *EDM2* signal changed to FALSE when output was not possible, and the monitoring time ended. *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C060	49248	EDM Error 23	The *EDM1* and *EDM2* signals are not valid during the initial status of the actuator. The *EDM1 and EDM2* signals changed to FALSE when output was not possible, and the monitoring time ended. *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C070	49264	EDM Error 31	The *EDM1* signal is not valid during the switched status of the actuator. The *EDM1* signal changed to TRUE when output was possible, and the monitoring time ended. *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C080	49280	EDM Error 32	The *EDM2* signal is not valid during the switched status of the actuator. The *EDM2* signal changed to TRUE when output was possible, and the monitoring time ended. *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C090	49296	EDM Error 33	The *EDM1* and *EDM2* signals are not valid during the switched status of the actuator. The *EDM1 and EDM2* signals changed to TRUE when output was possible, and the monitoring time ended. *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE
C111	49425	Init Error	*R_TRIG* was detected in the same cycle as *S_OutControl* and *Reset*. (There may be a programming error.) *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = TRUE

● FB-specific Status Codes (No Error)

DiagCode (hexa-decimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *S_EDM_Out* = FALSE *Error* = FALSE
8001	32769	Init	The FB was activated and started. Automatic resetting was FALSE at startup, so resetting is necessary. *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = FALSE
8010	32784	Output Disable	EDM control is OFF. The timer starts when this state is entered. *Ready* = TRUE *S_EDM_Out* = FALSE *Error* = FALSE
8000	32768	Output Enable	EDM control is ON. The timer starts when this state is entered. *Ready* = TRUE *S_EDM_Out* = TRUE *Error* = FALSE

SF_EmergencyStop

This safety FB monitors the input from an emergency stop button.

Instruction	Name	FB/FUN	Graphic expression
SF_EmergencyStop	Emergency Stop	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_EStopIn	SAFEBOOL	TRUE or FALSE	FALSE	A variable. This is a safety request input. FALSE: There is a request for a safety function. (Example: An emergency stop button was pressed.) TRUE: There is no request for a safety function. (Example: An emergency stop button was not pressed.)
S_StartReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_AutoReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
Reset	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.

Output Variables

Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_EStopOut	SAFEBOOL	TRUE or FALSE	FALSE	The safety function enable signal. FALSE: Disables the safety output. The safety function is operating. (Example: An emergency stop button was pressed, there was a reset request, or a valid internal error occurred.) TRUE: Enables the safety output. The safety function is not operating. (Example: An emergency stop button was not pressed or no valid internal error occurred.)
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

Precautions for Correct Use

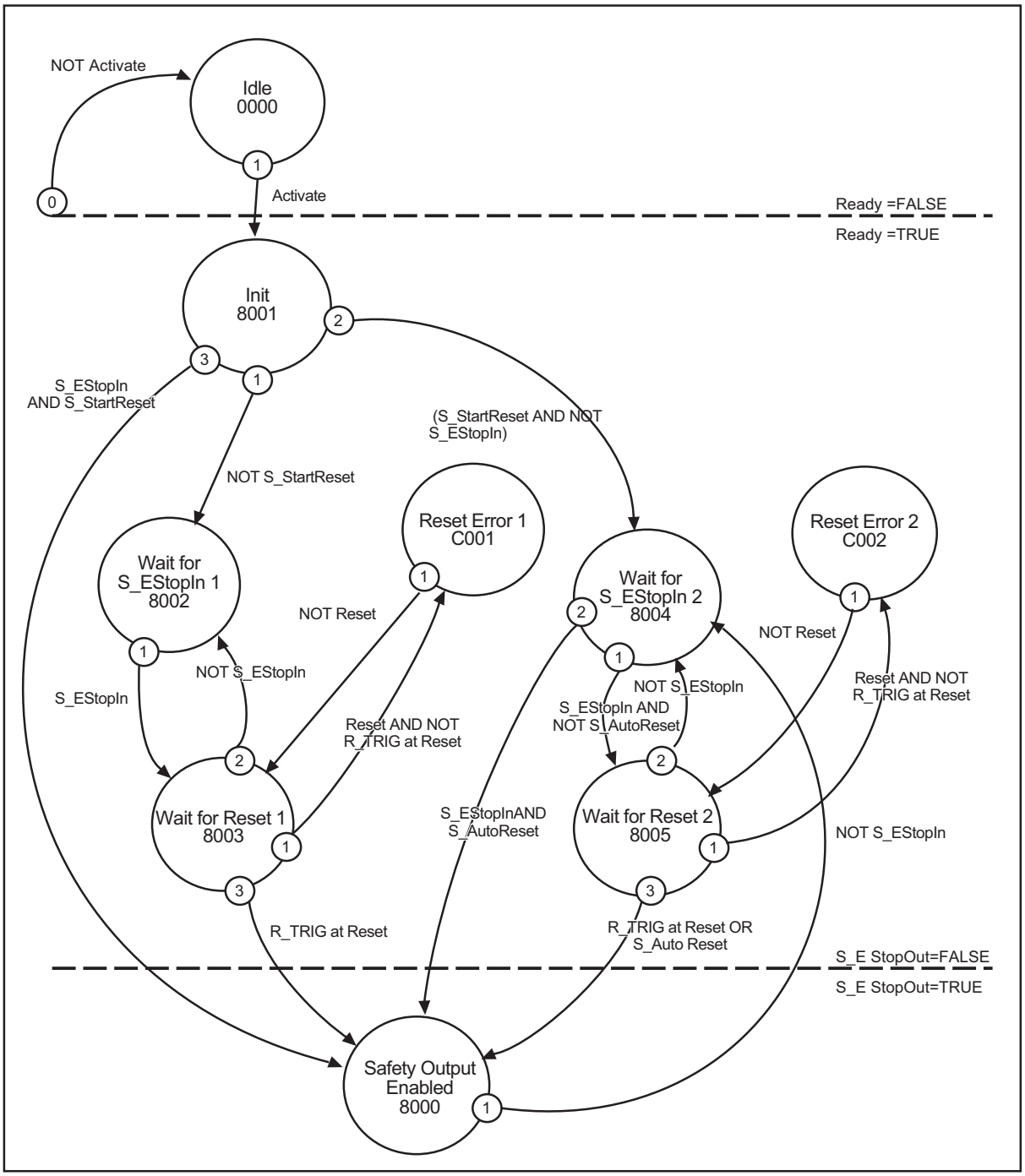
You must satisfy the following requirements, which are defined in ISO 13850 (EN 418).

- After activation of an actuator, an emergency stop device must operate to avoid or reduce any hazard by the best possible means.
- The emergency stop command must disable all other commands.
- A reset operation for any control device must be possible as the result of a manual operation on the control device itself. The machine must not restart until all activated control devices are reset manually, individually, and intentionally.

Function

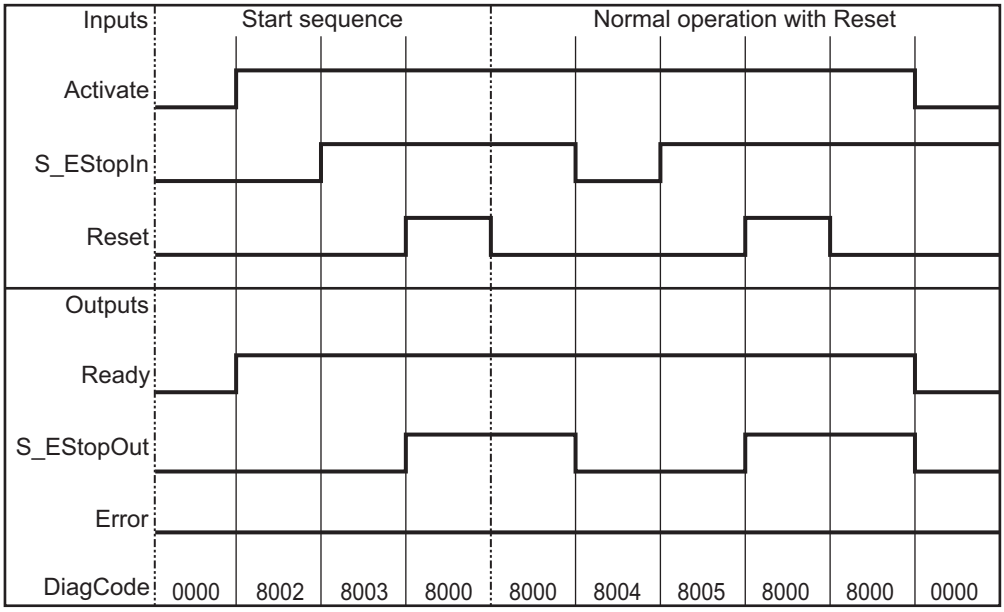
- When the *S_EStopIn* input is set to FALSE, *S_EStopOut* immediately changes to FALSE.
- *S_EStopOut* is reset to TRUE or waits to be reset only when the *S_EStopIn* input is set to TRUE. The conditions for waiting to be reset are determined by the defined *S_StartReset, S_AutoReset,* and *Reset* inputs.
- If *S_AutoReset* is TRUE, the confirmation operation is performed automatically.
- If *S_AutoReset* is FALSE, a change to TRUE in the *Reset* input must be made for enable confirmation.
- If *S_StartReset* is TRUE, the confirmation operation is performed automatically when the Safety CPU Unit first starts.
- If *S_StartReset* is FALSE, a change to TRUE in the *Reset* input must be made for enable confirmation.
- Activate the *S_StartReset* and *S_AutoReset* inputs only when you can ensure that no hazardous state will occur as the result of starting the Safety CPU Unit.
- You can use the SF_EmergencyStop instruction to monitor a single-channel or two-channel emergency stop button.
- The SF_EmergencyStop instruction automatically detects undetected changes to TRUE in the *Reset* input.

State Transition Diagram

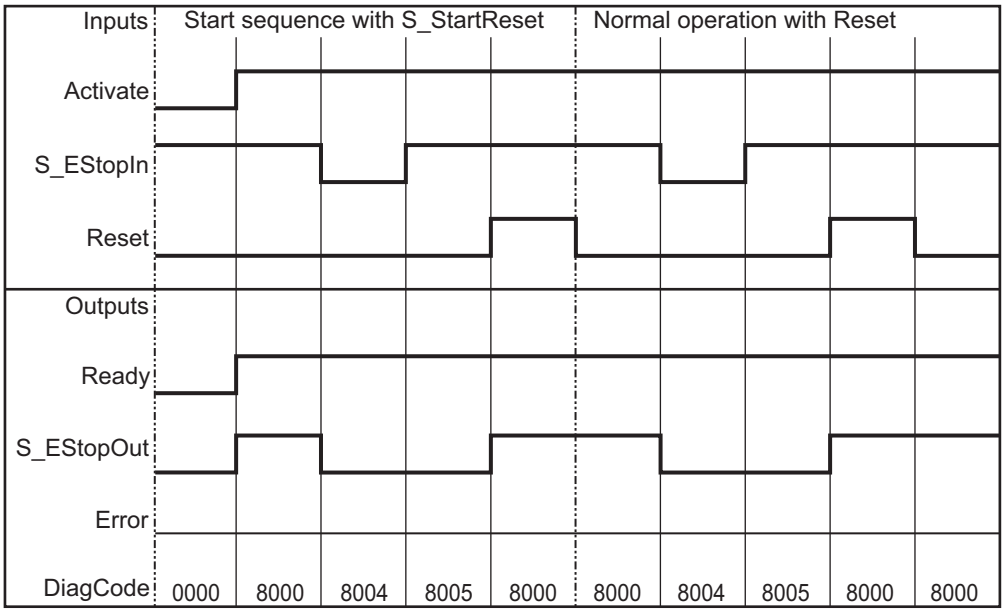


Note Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

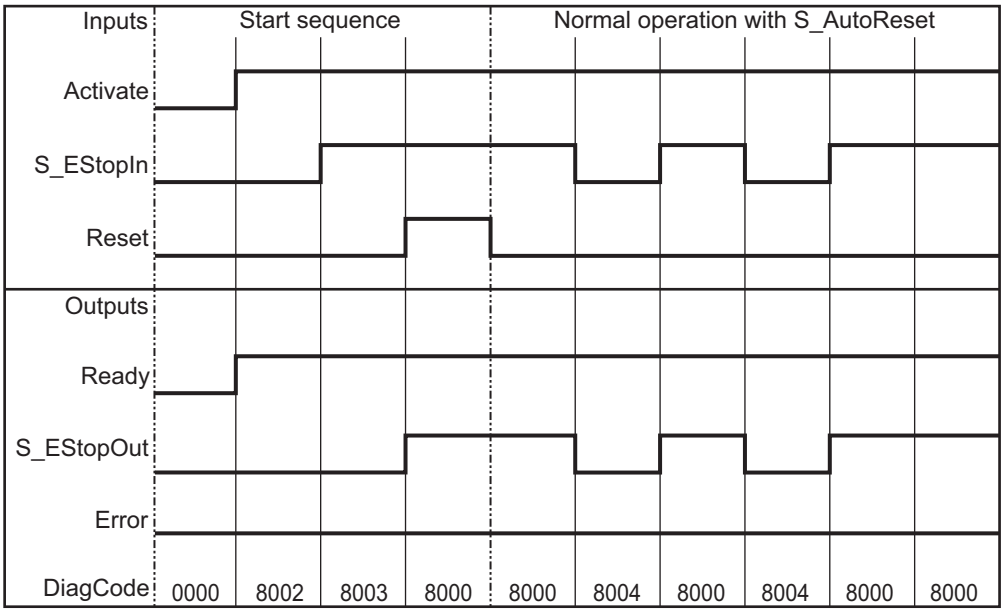
Timing Charts



S_StartReset = FALSE and S_AutoReset = FALSE: Start, reset, normal operation, safety request, and restart.



S_StartReset = TRUE and S_AutoReset = FALSE: Start, normal operation, safety request, and restart.



S_StartReset = FALSE and S_AutoReset = TRUE: Start, normal operation, safety request, and restart.

Instruction Execution Errors

Error Detected

This FB detects an undetected change to TRUE in the *Reset* input as an error.

Operation for Errors

- *S_EStopOut* is set to FALSE. If there is an undetected change to TRUE in the *Reset* input, the *DiagCode* output gives the relevant error code and the *Error* output is set to TRUE.
- To reset the error, you must set *Reset* to FALSE.

FB-specific Error Codes

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C001	49153	Reset Error 1	When the Wait for Reset 1 state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_EStopOut* = FALSE *Error* = TRUE
C002	49154	Reset Error 2	When the Wait for Reset 2 state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_EStopOut* = FALSE *Error* = TRUE

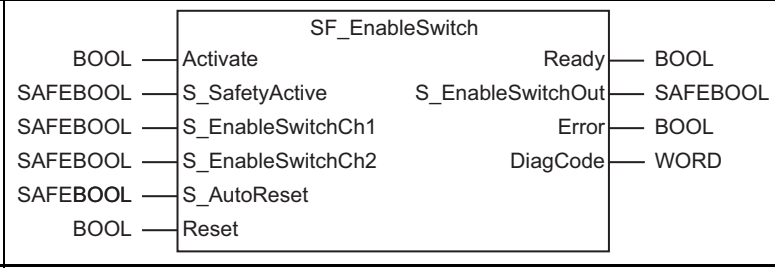
FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *S_EStopOut* = FALSE *Error* = FALSE
8001	32769	Init	The FB detected an activate signal and the FB is active. See if the *S_StartReset* input is required. *Ready* = TRUE *S_EStopOut* = FALSE *Error* = FALSE
8002	32770	Wait for S_EStopIn 1	The FB is waiting for *S_EStopIn* to change to TRUE. Make sure that *Reset* is FALSE. *Ready* = TRUE *S_EStopOut* = FALSE *Error* = FALSE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8003	32771	Wait for Reset 1	*S_EStopIn* is TRUE. The FB is waiting for Reset to change to TRUE. *Ready* = TRUE *S_EStopOut* = FALSE *Error* = FALSE
8004	32772	Wait for S_EStopIn 2	A safety request was detected. Make sure that *Reset* is FALSE. The FB is waiting for *S_EStopIn* to change to TRUE. *Ready* = TRUE *S_EStopOut* = FALSE *Error* = FALSE
8005	32773	Wait for Reset 2	*S_EStopIn is TRUE.* Check *S_AutoReset*. Or, the FB is waiting for *Reset* to change to TRUE. *Ready* = TRUE *S_EStopOut* = FALSE *Error* = FALSE
8000	32768	Safety Output Enabled	*S_EStopIn* is TRUE and *S_EStopOut* is TRUE (function mode). *Ready* = TRUE *S_EStopOut* = TRUE *Error* = FALSE

SF_EnableSwitch

This safety FB supports stopping a safety protection function that uses an enable switch.

Instruction	Name	FB/FUN	Graphic expression
SF_EnableSwitch	Enable Switch	FB	

Variables

Input Variables

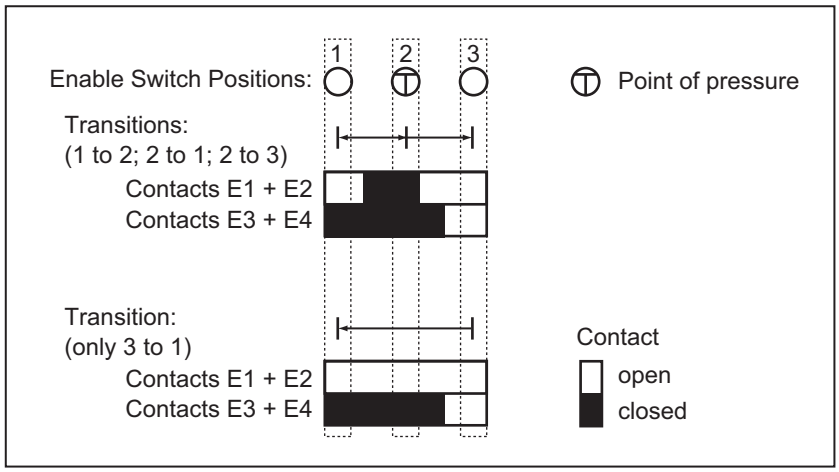
Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_SafetyActive	SAFEBOOL	TRUE or FALSE	FALSE	A constant or a variable. It confirms safe mode (motion speed limit, power limit, or motion range limit). FALSE: Turns OFF safe mode. TRUE: Turns ON safe mode.
S_EnableSwitchCh1	SAFEBOOL	TRUE or FALSE	FALSE	A variable. The input signal from connected enable switches E1 and E2. FALSE: The connected switches are open. TRUE: The connected switches are closed.
S_EnableSwitchCh2	SAFEBOOL	TRUE or FALSE	FALSE	A variable. The input signal from connected enable switches E3 and E4. FALSE: The connected switches are open. TRUE: The connected switches are closed.
S_AutoReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
Reset	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.

Output Variables

Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_EnableSwitchOut	SAFEBOOL	TRUE or FALSE	FALSE	A safety-related output that indicates that the safety protection function is stopped. FALSE: Disables stopping the safety protection function. TRUE: Enables stopping the safety protection function.
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

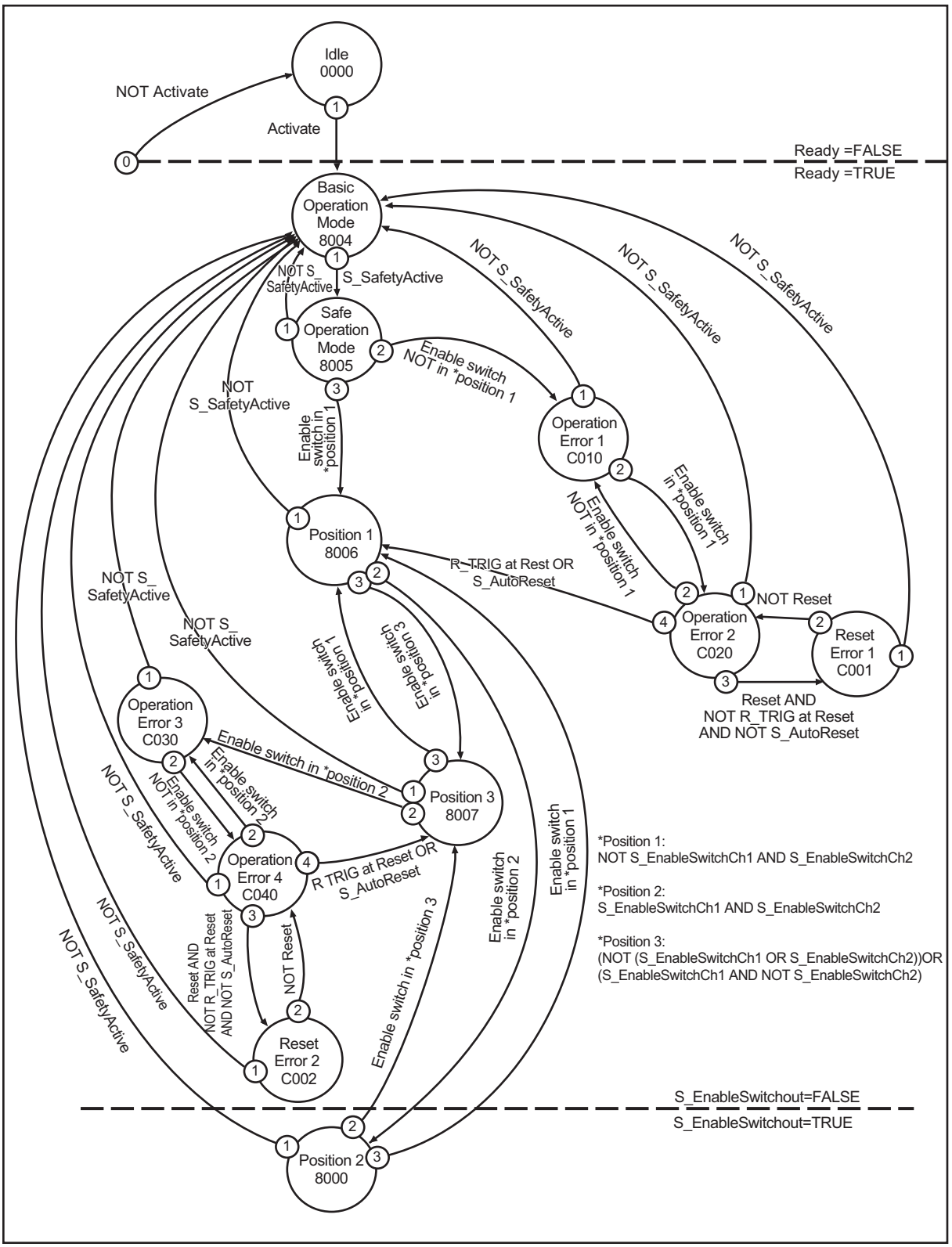
Function

- This FB supports stopping a safety protection function that uses an enable switch when a suitable operation mode is started. However, handle the related operation mode (motion speed limit, power limit, or motion range limit) outside of the SF_EnableSwitch instruction.
- This FB evaluates enable switch signals for three positions (IEC 60204 section 10.9).
- The *S_EnableSwitchCh1* and *S_EnableSwitchCh2* input parameters are processed as shown below for the E1 to E4 input signal levels.



- You must connect the signals from E1 and E2 to the *S_EnableSwitchCh1* parameter. You must also connect the signals from E3 and E4 to the *S_EnableSwitchCh2* parameter. The FB detects the position of the enable switch with this signal sequence. However, the transition from position 2 to position 3 is sometimes different from the one that is shown in the figure.
- The FB can detect the direction in which the switch changed (from position 1 to position 2 or from position 3 to position 2) by using the signal sequences that are defined for the enable switch inputs. The FB enables stopping the safety protection function only when the switch changes from position 1 to position 2. You cannot use any other direction or switch position to enable stopping the safety protection function. This function is based on section 10.9 of IEC 60204. (The type of brackets that is specified in EN 60204 is different).
- Because it is based on section 10.9 of IEC 60204, it is necessary to use a suitable switching device. You must also confirm that a suitable operation mode is selected in the relevant application. (In this operation mode, you must disable automatic operation with a suitable means.)
- An operation mode selection switch is normally used to select the operation mode to move the machine to the safe state with the SF_ModeSelector and SF_SafeRequest instructions.
- The SF_EnableSwitch instruction confirms safe mode with the value of *S_SafetyActive*. If implementation does not require confirmation of safe mode in the application, connect a constant TRUE signal to *S_SafetyActive*.
- Activate the *S_StartReset* input only when you can ensure that no hazardous state will occur as the result of starting the Safety CPU Unit.

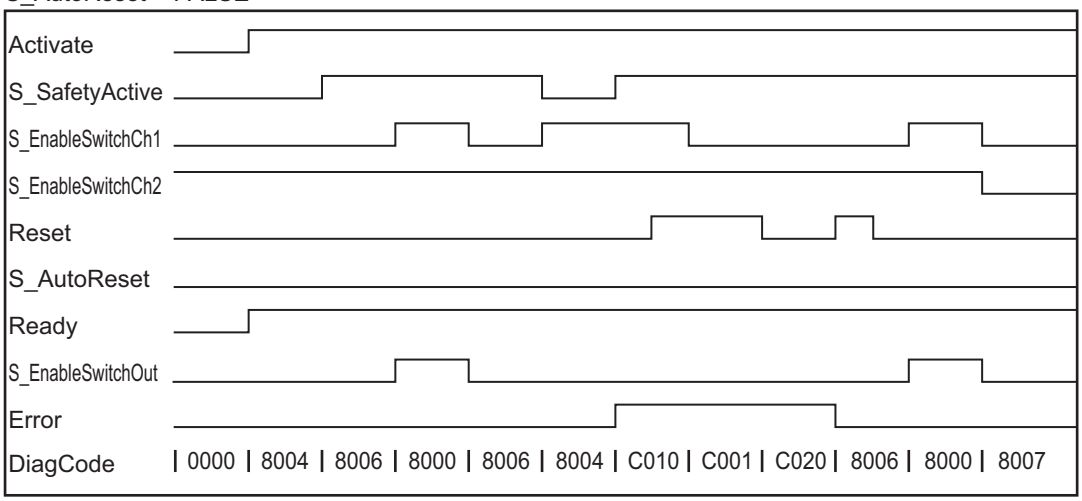
State Transition Diagram



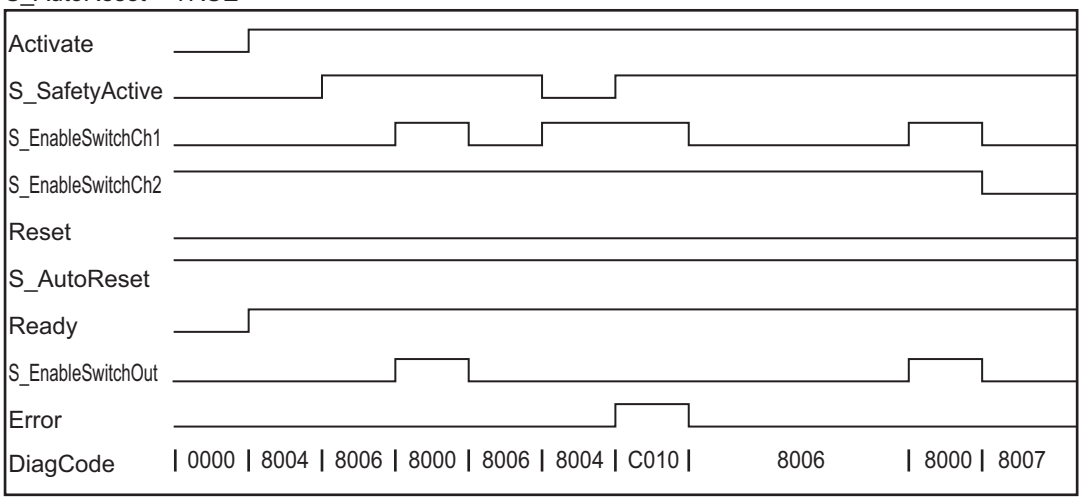
Note Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

Timing Charts

S_AutoReset = FALSE



S_AutoReset = TRUE



Instruction Execution Errors

Error Detected

The following conditions force a transition to an error state.

- When an undetected change to TRUE in the *Reset* input is detected in the Operation Error 2 or Operation Error 4 state
- When the switch position is not valid

Operation for Errors

- If an error occurs, the *S_EnableSwitchOut* safety output changes to FALSE and the safe state is maintained. As opposed to other FBs, the reset error state is maintained when *Reset* is FALSE, including when *S_SafetyActive* is also FALSE.
- After the error is reset, the enable switch must be set to the initial position that was defined for the process before the enable switch can set the *S_EnableSwitchOut* output to TRUE. If *S_AutoReset* is FALSE, *Reset* must be changed from FALSE to TRUE.

FB-specific Error Codes

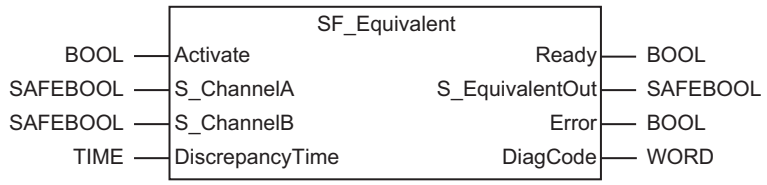
DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C001	49153	Reset Error 1	When the Operation Error 2 state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_EnableSwitchOut* = FALSE *Error* = TRUE
C002	49154	Reset Error 2	When the Operation Error 4 state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_EnableSwitchOut* = FALSE *Error* = TRUE
C010	49168	Operation Error 1	The enable switch was not set in position 1 when *S_SafetyActive* was started. *Ready* = TRUE *S_EnableSwitchOut* = FALSE *Error* = TRUE
C020	49184	Operation Error 2	The enable switch was set in position 1 after C010. *Ready* = TRUE *S_EnableSwitchOut* = FALSE *Error* = TRUE
C030	49200	Operation Error 3	The enable switch was set in position 2 from position 3. *Ready* = TRUE *S_EnableSwitchOut* = FALSE *Error* = TRUE
C040	49216	Operation Error 4	The enable switch was not set in position 2 after C030. Ready = TRUE *S_EnableSwitchOut* = FALSE *Error* = TRUE

● FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *S_EnableSwitchOut* = FALSE *Error* = FALSE
8004	32772	Basic Operation Mode	Safe operation mode is OFF. *Ready* = TRUE *S_EnableSwitchOut* = FALSE *Error* = FALSE
8005	32773	Safe Operation Mode	Safe operation mode is ON. *Ready* = TRUE *S_EnableSwitchOut* = FALSE *Error* = FALSE
8006	32774	Position 1	Safe operation mode is ON and the enable switch is in position 1. *Ready* = TRUE *S_EnableSwitchOut* = FALSE *Error* = FALSE
8007	32775	Position 3	Safe operation mode is ON and the enable switch is in position 3. *Ready* = TRUE *S_EnableSwitchOut* = FALSE *Error* = FALSE
8000	32768	Position 2	Safe operation mode is ON and the enable switch is in position 2. *Ready* = TRUE *S_EnableSwitchOut* = TRUE *Error* = FALSE

SF_Equivalent

This safety FB monitors the discrepancy time for two equivalent SAFEBOOL inputs.

Instruction	Name	FB/FUN	Graphic expression
SF_Equivalent	Equivalent	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_ChannelA	SAFEBOOL	TRUE or FALSE	FALSE	A variable. It functions as input A for a logic connection. FALSE: Input A is open. TRUE: Input A is closed.
S_ChannelB	SAFEBOOL	TRUE or FALSE	FALSE	A variable. It functions as input B for a logic connection. FALSE: Input B is open. TRUE: Input B is closed.
Discrepancy-Time	TIME	Depends on data type.	T#0ms	A constant. It sets the maximum monitoring time for discrepancy between two inputs.

Output Variables

Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_EquivalentOut	SAFEBOOL	TRUE or FALSE	FALSE	The safety output. FALSE: At least one of the signals is FALSE or the state did not change within the monitoring time. TRUE: The two input signals are active and the state changed within the monitoring time.
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

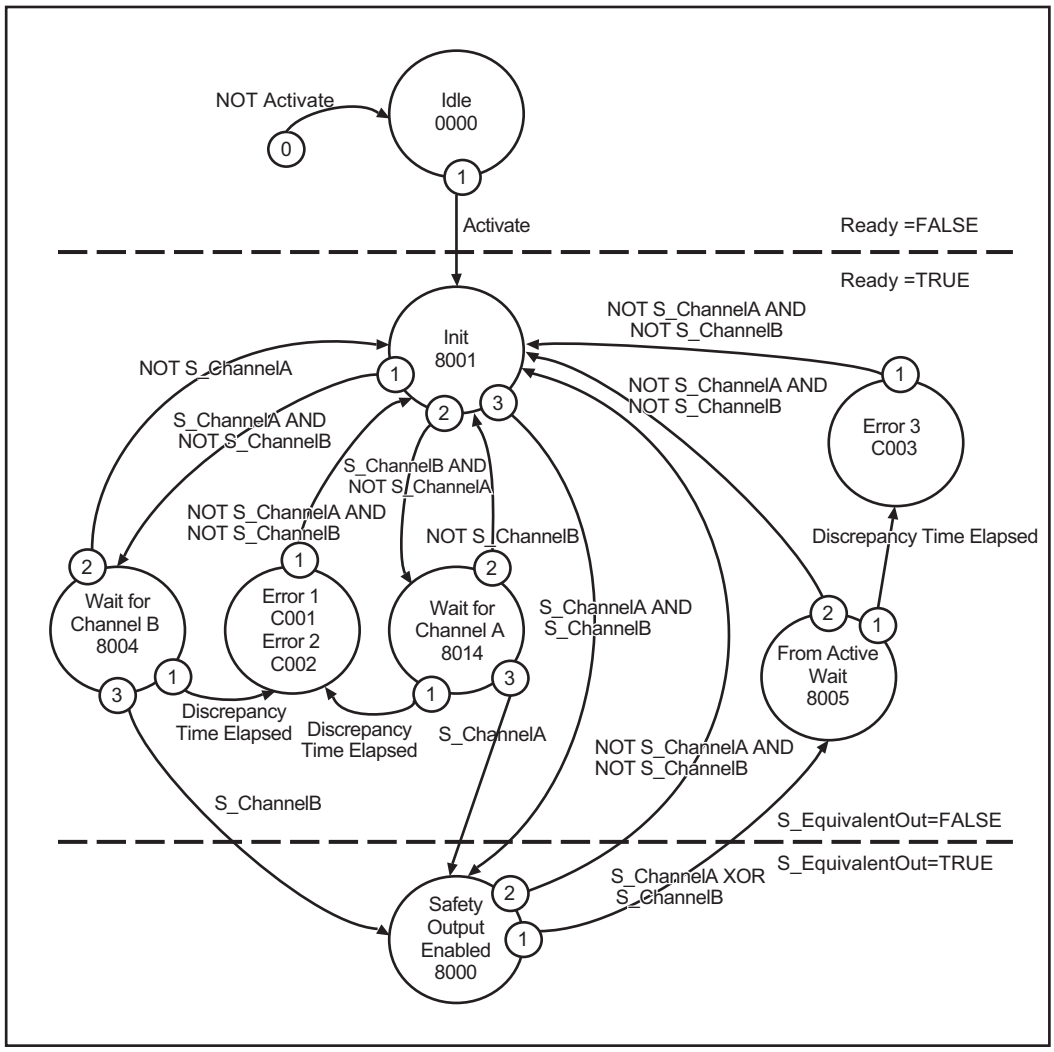
Function

- This FB monitors the time that two equivalent SAFEBOOL inputs are not the same and converts the inputs to one SAFEBOOL output.
- *S_ChannelA* and *S_ChannelB* are dependent on each other. The evaluation result for both channels is output.
- When the input for one channel changes so that the two channel inputs are no longer the same, the FB starts monitoring the discrepancy time. An error occurs if a change does not occur that makes them the same again within the monitoring time.
- *S_EquivalentOut* is TRUE if the conditions for both changes are met and FALSE if the conditions are not met.
- If an error occurs, make the inputs for both channels inactive (make *S_ChannelA* and *S_ChannelA* FALSE) to reset the FB.
- Set *DiscrepancyTime* to a value that is longer than the safety task period.
  Refer to the *NX-series Safety Control Unit User's Manual* (Cat. No. Z930) for application methods for *DiscrepancyTime*.

Precautions for Correct Use

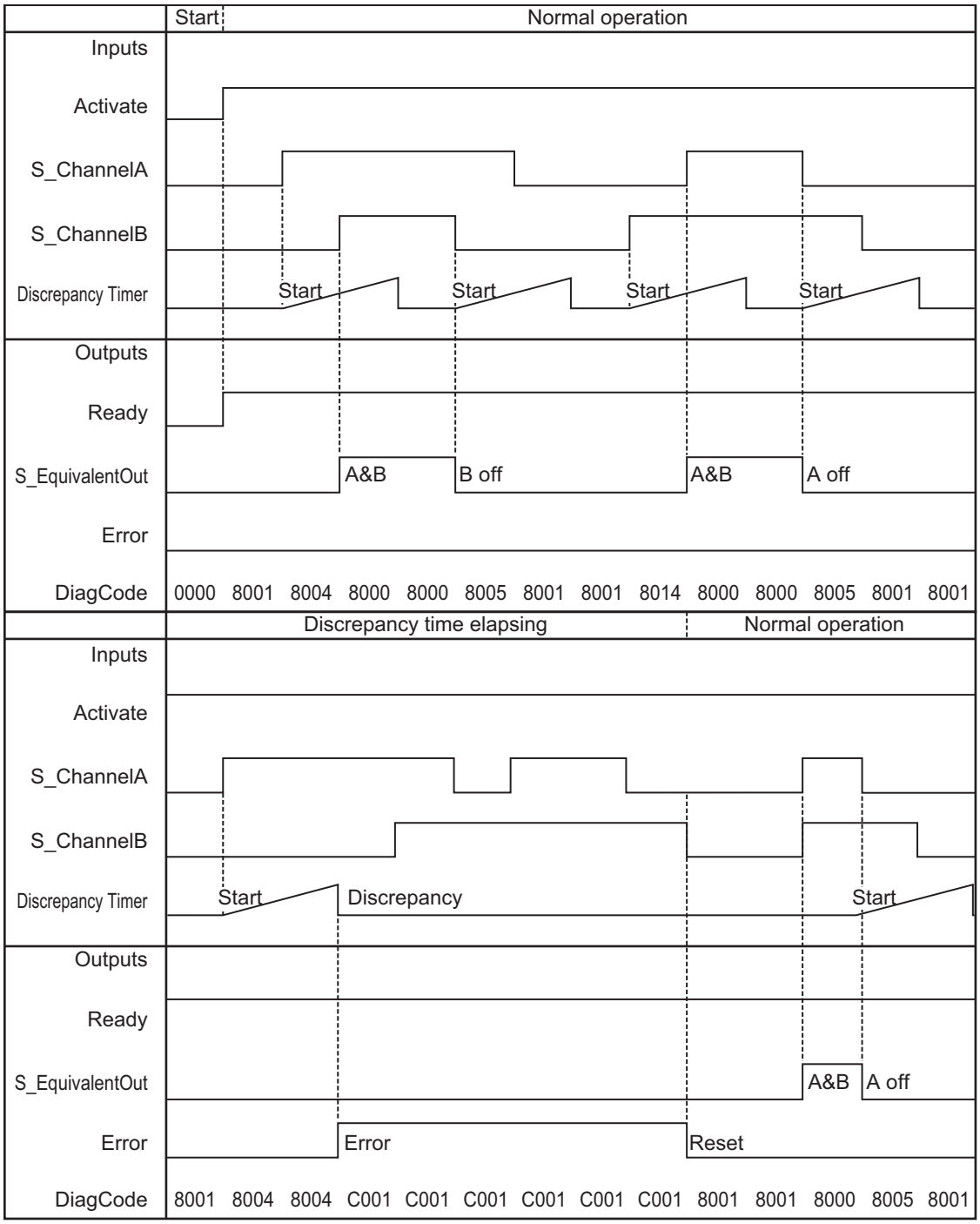
This FB does not have a restart interlock. You must connect it to a FB that has a restart interlock.

State Transition Diagram



Note Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

Timing Charts



Instruction Execution Errors

Error Detected

The discrepancy time between *S_ChannelA* and *S_ChannelB* is monitored when either of them changes to TRUE or FALSE.

Operation for Errors

- If an error is detected, *S_EquivalentOut* changes to FALSE and *Error* changes to TRUE. *DiagCode* shows the error state.
- If an error occurs in an input, make the inputs for both channels inactive (make *S_ChannelA* and *S_ChannelA* FALSE) to reset the FB.

FB-specific Error Codes

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C001	49153	Error 1	*S_ChannelB* did not change to TRUE within *MonitoringTime* in the Wait for Channel B state (8004). *Ready* = TRUE *S_EquivalentOut* = FALSE *Error* = TRUE
C002	49154	Error 2	*S_ChannelA* did not change to TRUE within *MonitoringTime* in the Wait for Channel A state (8014). *Ready* = TRUE *S_EquivalentOut* = FALSE *Error* = TRUE
C003	49155	Error 3	The input did not change within the monitoring time while changing from the From Active Wait (8005) to the Init (8001) state. *S_ChannelA* or *S_ChannelB* did not change to FALSE within the monitoring time after the other input changed to FALSE. *Ready* = TRUE *S_EquivalentOut* = FALSE *Error* = TRUE

FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *S_EquivalentOut* = FALSE *Error* = FALSE
8001	32769	Init	The FB detected an activate signal and the FB is active. *Ready* = TRUE *S_EquivalentOut* = FALSE *Error* = FALSE
8000	32768	Safety Output Enabled	An input changed to TRUE in Equivalent Mode. *Ready* = TRUE *S_EquivalentOut* = TRUE *Error* = FALSE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8004	32772	Wait for Channel B	*S_ChannelA* changed to TRUE, the discrepancy time timer started operation, and the FB is waiting for *S_ChannelB* to change to TRUE. *Ready* = TRUE *S_EquivalentOut* = FALSE *Error* = FALSE
8014	32788	Wait for Channel A	*S_ChannelB* changed to TRUE, the discrepancy time timer started operation, and the FB is waiting for *S_ChannelA* to change to TRUE. *Ready* = TRUE *S_EquivalentOut* = FALSE *Error* = FALSE
8005	32773	From Active Wait	One of the channels changed to FALSE, the discrepancy time timer started operation, and the FB is waiting for the other channel to change to FALSE. *Ready* = TRUE *S_EquivalentOut* = FALSE *Error* = FALSE

SF_ESPE

This safety FB monitors electro-sensitive protective equipment (ESPE). ESPE includes light curtains, laser scanners, etc.

Instruction	Name	FB/FUN	Graphic expression
SF_ESPE	Electro-Sensitive Protective Equipment (ESPE)	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_ESPE_In	SAFEBOOL	TRUE or FALSE	FALSE	A variable. This is a safety request input. FALSE: There is a request for a safety function. TRUE: There is no request for a safety function. If the ESPE is used as a stopping device, the safety control system must detect a short shutoff (80 ms min. according to IEC 61496-1) with a sensor.
S_StartReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_AutoReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
Reset	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.

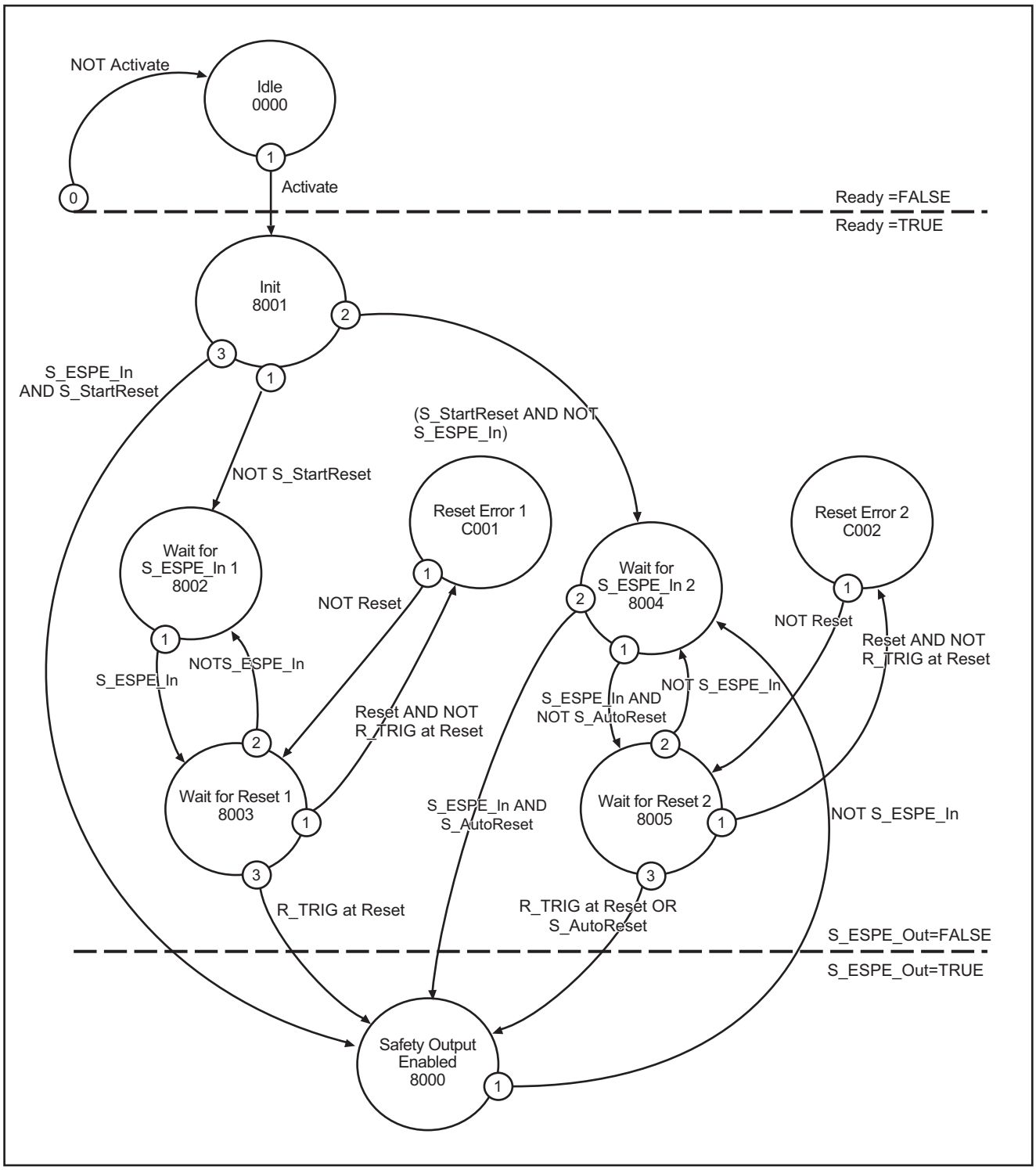
Output Variables

Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_ESPE_Out	SAFEBOOL	TRUE or FALSE	FALSE	The safety function enable signal. FALSE: Disables the safety output. There is a request for a safety-related response (example: a reset request or an internal error). TRUE: Enables the safety output. There is no request for a safety-related response.
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

Function

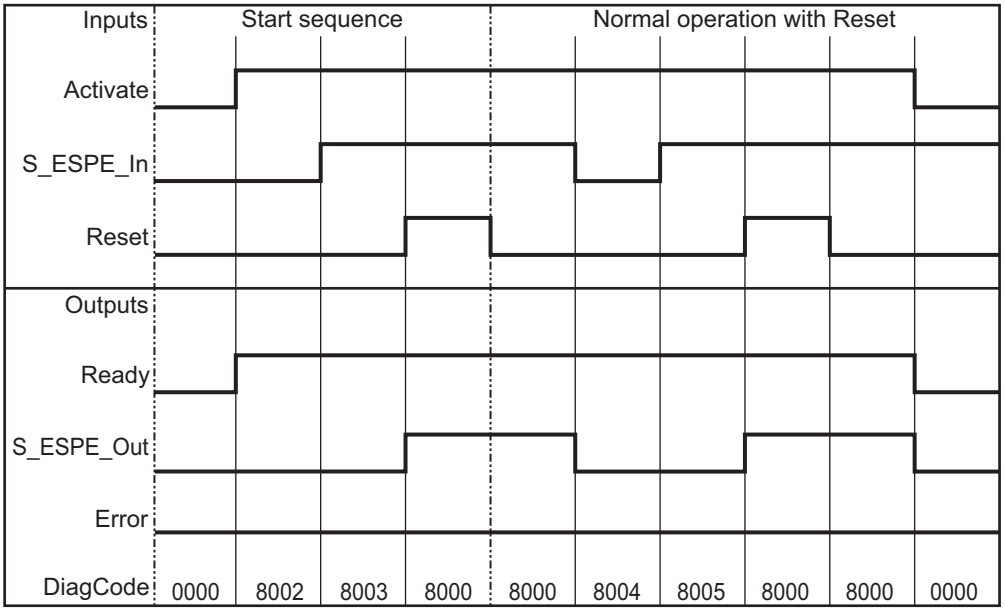
- This FB monitors electro-sensitive protective equipment (ESPE). The function is the same as that of the SF_EmergensyStop instruction. As soon as the *S_ESPE_In* input is set to FALSE, the *S_ESPE_Out* output is set to FALSE. If the *S_ESPE_In* input is set to TRUE and the function is reset, the *S_ESPE_Out* output signal is set to TRUE. Enabling the reset is determined by the defined *S_StartReset*, *S_AutoReset*, and *Reset* inputs.
- If *S_AutoReset* is TRUE, the confirmation operation is performed automatically.
- If *S_AutoReset* is FALSE, a change to TRUE in the *Reset* input must be made for enable confirmation.
- If *S_StartReset* is TRUE, the confirmation operation is performed automatically when the Safety CPU Unit first starts.
- If *S_StartReset* is FALSE, a change to TRUE in the *Reset* input must be made for enable confirmation.
- Activate the *S_StartReset* and *S_AutoReset* inputs only when you can ensure that no hazardous state will occur as the result of starting the Safety CPU Unit.
- ESPE must be selected for the category that is required according to product standards IEC/EN 61496-1, IEC/EN 61496-2, IEC/EN 61496-3, and EN ISO 13849-1.

State Transition Diagram

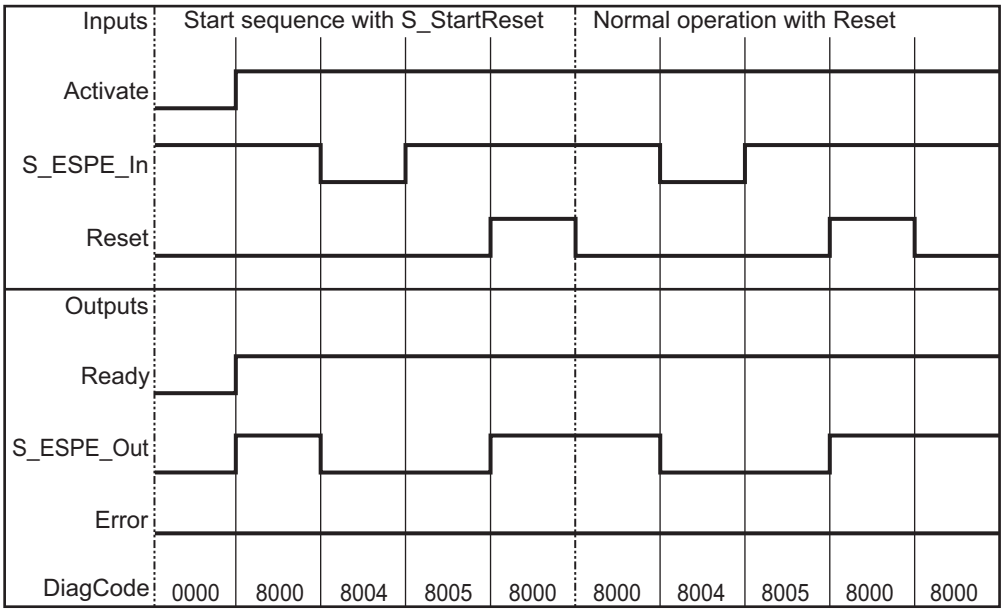


Note Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

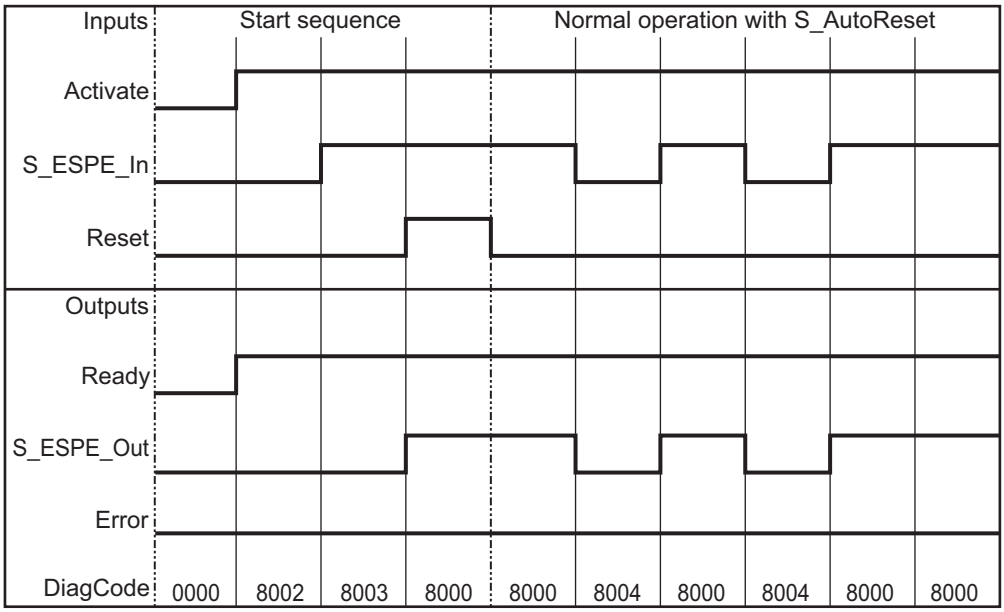
Timing Charts



S_StartReset = FALSE and S_AutoReset = FALSE: Start, reset, normal operation, safety request, and restart.



S_StartReset = TRUE and S_AutoReset = FALSE: Start, normal operation, safety request, and restart.



S_StartReset = FALSE and S_AutoReset = TRUE: Start, normal operation, safety request, and restart.

Instruction Execution Errors

Error Detected

This FB detects an undetected change to TRUE in the *Reset* input as an error.

Operation for Errors

- *S_ESPE_Out* is set to FALSE. If there is an undetected change to TRUE in the *Reset* input, the *DiagCode* output gives the relevant error code and the *Error* output is set to TRUE.
- To reset the error, you must set *Reset* to FALSE.

FB-specific Error Codes

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C001	49153	Reset Error 1	When the Wait for Reset 1 state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_ESPE_Out* = FALSE *Error* = TRUE
C002	49154	Reset Error 2	When the Wait for Reset 2 state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_ESPE_Out* = FALSE *Error* = TRUE

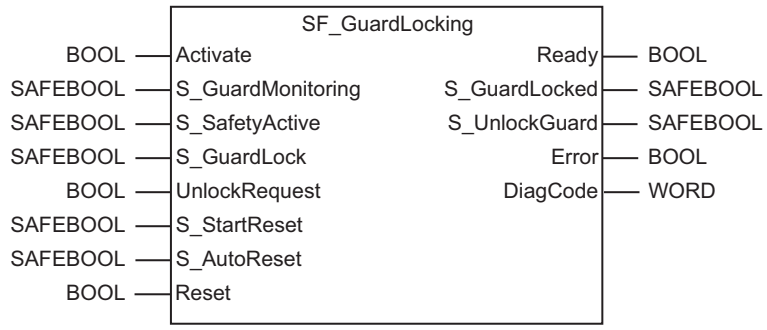
FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *S_ESPE_Out* = FALSE *Error* = FALSE
8001	32769	Init	The FB is active. The FB is already started. See if *S_StartReset* is required. *Ready* = TRUE *S_ESPE_Out* = FALSE *Error* = FALSE
8002	32770	Wait for S_ESPE_In 1	The FB is active. Make sure that *Reset* is FALSE. The FB is waiting for *S_ESPE_In* to change to TRUE. *Ready* = TRUE *S_ESPE_Out* = FALSE *Error* = FALSE
8003	32771	Wait for Reset 1	The FB is active. *S_ESPE_In* is *TRUE.* The FB is waiting for *Reset* to change to TRUE. *Ready* = TRUE *S_ESPE_Out* = FALSE *Error* = FALSE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8004	32772	Wait for S_ESPE_In 2	The FB is active. A safety request was detected. Make sure that *Reset* is FALSE. The FB is waiting for *S_ESPE_In* to change to TRUE. *Ready* = TRUE *S_ESPE_Out* = FALSE *Error* = FALSE
8005	32773	Wait for Reset 2	The FB is active. *S_ESPE_In* is TRUE. Check *S_AutoReset.* Or, the FB is waiting for *Reset* to change to TRUE. *Ready* = TRUE *S_ESPE_Out* = FALSE *Error* = FALSE
8000	32768	Safety Output Enabled	The FB is active. *S_ESPE_In* is TRUE and *S_ESPE_Out* is TRUE (function mode). *Ready* = TRUE *S_ESPE_Out* = TRUE *Error* = FALSE

SF_GuardLocking

This safety FB controls entry to a hazardous area with a four-state interlock guard with a guard lock.

Instruction	Name	FB/FUN	Graphic expression
SF_GuardLocking	Safety Guard Interlocking with Locking	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_GuardMonitoring	SAFEBOOL	TRUE or FALSE	FALSE	A variable. It monitors the guard interlock. FALSE: The guard is open. TRUE: The guard is closed.
S_SafetyActive	SAFEBOOL	TRUE or FALSE	FALSE	A variable. It monitors the velocity or the state of the hazardous area (EDM) based on the safe time OFF delay, etc. FALSE: The mechanical device is in a non-safe state. TRUE: The mechanical device is in a safe state.
S_GuardLock	SAFEBOOL	TRUE or FALSE	FALSE	A variable. It gives the status of the mechanical guard lock. FALSE: The guard is not locked. TRUE: The guard is locked.
UnlockRequest	BOOL	TRUE or FALSE	FALSE	A variable. It indicates operator intervention (i.e., a request to unlock the guard). FALSE: There is no request. TRUE: There is a request.
S_StartReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_AutoReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
Reset	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.

Output Variables

Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_GuardLocked	SAFEBOOL	TRUE or FALSE	FALSE	Connect this output to actuator that is the hazard source inside the guard. The safety output changes to FALSE when there is a request to unlock the guard. FALSE: A non-safe state exists. TRUE: A safe state exists.
S_UnlockGuard	SAFEBOOL	TRUE or FALSE	FALSE	The guard unlock signal. FALSE: Closes the guard. TRUE: Releases the guard.
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

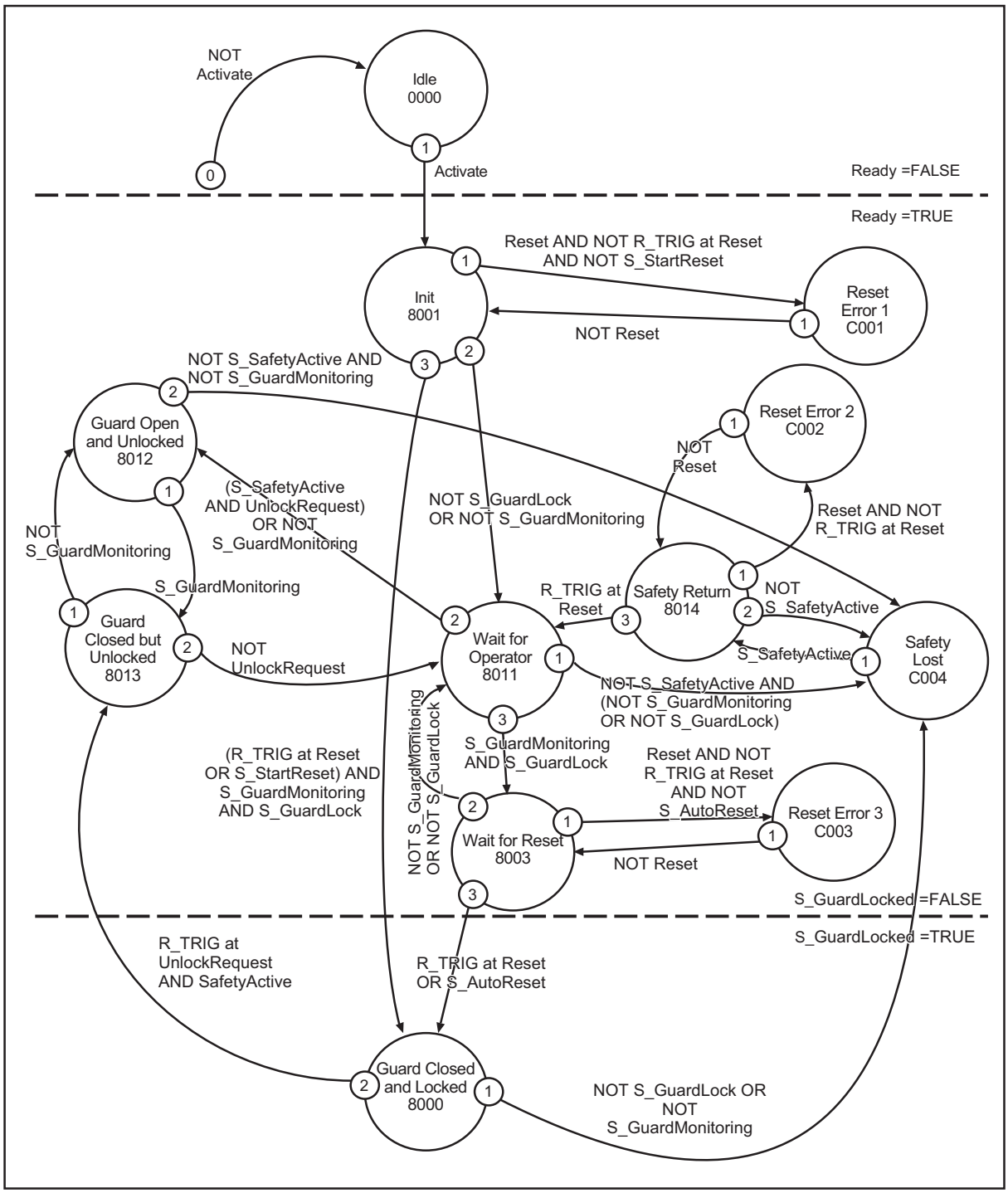
Function

- This FB controls a guard lock. It monitors the guard and lock position. You can use this FB with a mechanical lock switch.
- The operator requests access to a hazardous area. The guard can be unlocked only when the hazardous area is in a safe state. If the guard is closed, it can be locked. If the guard is closed and locked, the mechanical device can be started. An open guard or an unlocked guard is detected as a situation that has a serious impact on safety.
- Activate the *S_StartReset* and *S_AutoReset* inputs only when you can ensure that no hazardous state will occur as the result of starting the Safety CPU Unit.

● Operating Sequence

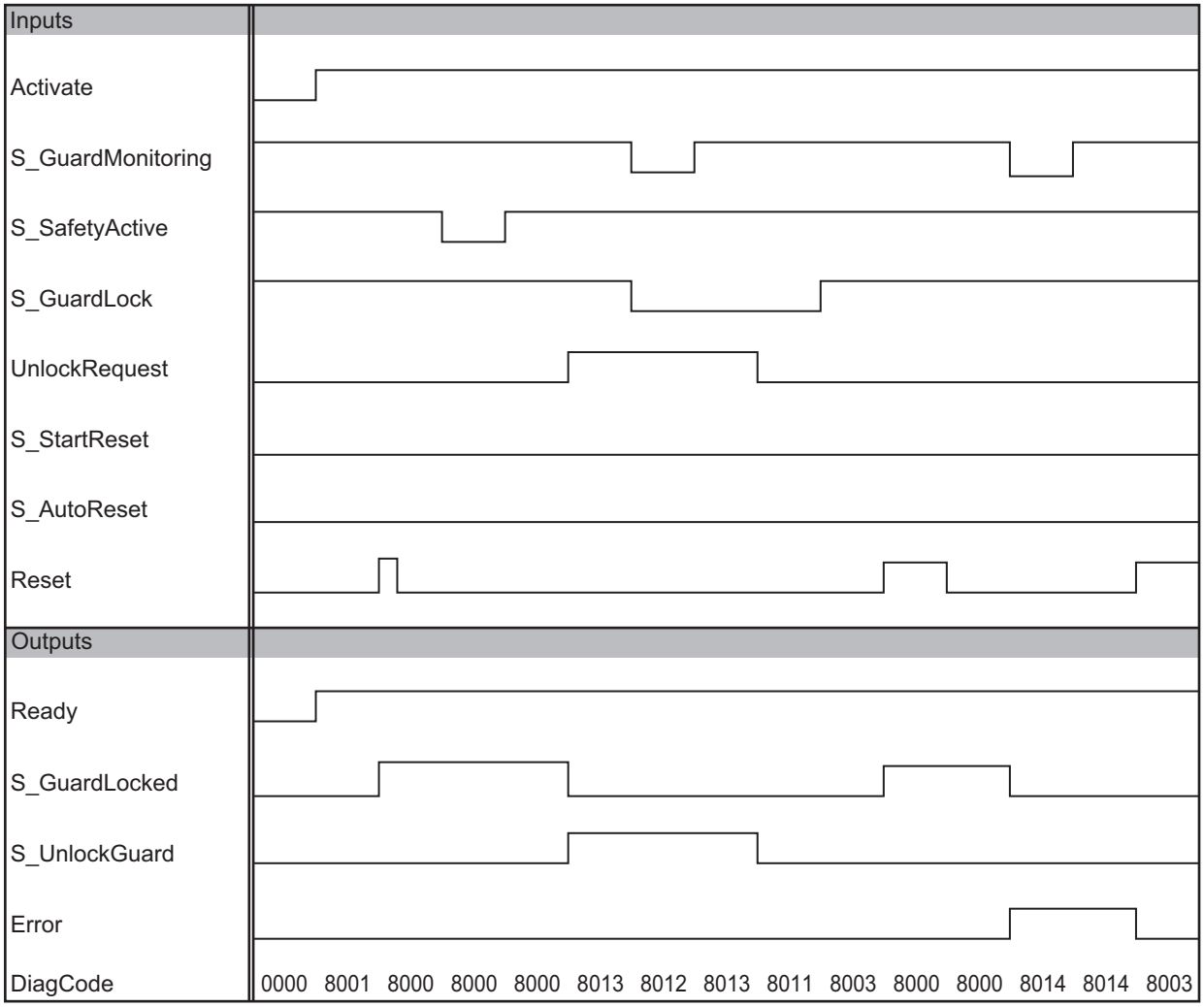
1	External	A request is made to place the hazardous area in the safe state. (This is not included in this FB.)
2	Input	Feedback is given from the relevant hazardous area that it is in the safe state. (Use *S_SafetyActive*.)
3	Input	Operator requests unlocking the guard. (Use *UnlockRequest*.)
4	Output	Opening the guard is enabled. (Use *S_UnlockGuard*.)
5	Input	The guard is unlocked. (Use *S_GuardLock*.) Opening the guard is enabled (*S_GuardLocked = FALSE*). The operator opens the guard.
6	Input	Monitoring the status of the guard through *S_GuardMonitoring* (signal for closing the guard again) starts.
7	Input	The operator provides feedback (*Reset*) to start the hazardous area again.
8	Output	The guard is locked (*S_UnlockGuard*).
9	Input	The locking of the guard is confirmed (*S_GuardLock*).
10	Output	The hazardous area is made operational again (*S_GuardLocked* = TRUE).
11	External	Operation in the hazardous area is restarted.

State Transition Diagram



Note Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

Timing Charts



Instruction Execution Errors

Error Detected

- Undetected changes to TRUE in the *Reset* input are detected. Errors are detected with the guard switch.
- An error is detected when safety is compromised. Either the guard was opened or unlocked.

Operation for Errors

- When an error occurs, the *S_GuardLocked* and *S_UnlockGuard* outputs are set to FALSE, the *DiagCode* output gives the relevant error code, and the *Error* output is set to TRUE.
- An error must be acknowledged by changing the *Reset* input to TRUE.

FB-specific Error Codes

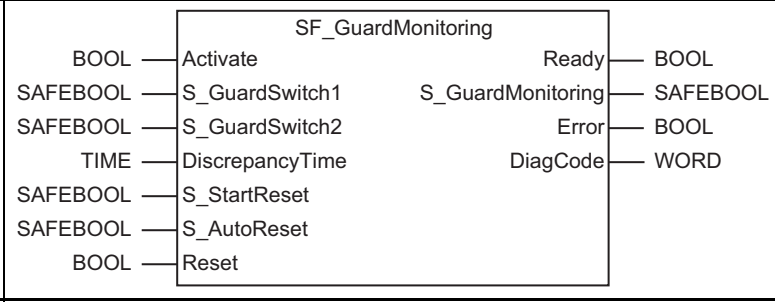
DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C001	49153	Reset Error 1	When the Init state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_GuardLocked* = FALSE *S_UnlockGuard* = FALSE *Error* = TRUE
C002	49154	Reset Error 2	When the Safety Lost state (C004) was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_GuardLocked* = FALSE *S_UnlockGuard* = FALSE *Error* = TRUE
C003	49155	Reset Error 3	When the Wait for Reset state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_GuardLocked* = FALSE *S_UnlockGuard* = FALSE *Error* = TRUE
C004	49156	Safety Lost	Safety was compromised. Either the guard was opened or unlocked. *Ready* = TRUE *S_GuardLocked* = FALSE *S_UnlockGuard* = FALSE *Error* = TRUE

● FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *S_GuardLocked* = FALSE *S_UnlockGuard* = FALSE *Error* = FALSE
8000	32768	Guard Closed and Locked	The guard is locked. *Ready* = TRUE *S_GuardLocked* = TRUE *S_UnlockGuard* = FALSE *Error* = FALSE
8001	32769	Init	The FB was activated and started. *Ready* = TRUE *S_GuardLocked* = FALSE *S_UnlockGuard* = FALSE *Error* = FALSE
8003	32771	Wait for Reset	The door is closed and locked and the FB is waiting for the operator to reset the function. *Ready* = TRUE *S_GuardLocked* = FALSE *S_UnlockGuard* = FALSE *Error* = FALSE
8011	32785	Wait for Operator	The FB is waiting for the operator to request unlocking the guard or resetting the function. *Ready* = TRUE *S_GuardLocked* = FALSE *S_UnlockGuard* = FALSE *Error* = FALSE
8012	32786	Guard Open and Unlocked	The guard is unlocked and open. *Ready* = TRUE *S_GuardLocked* = FALSE *S_UnlockGuard* = TRUE *Error* = FALSE
8013	32787	Guard Closed but Unlocked	The guard is unlocked but closed. *Ready* = TRUE *S_GuardLocked* = FALSE *S_UnlockGuard* = TRUE *Error* = FALSE
8014	32788	Safety Return	The *S_SafetyActive* signal was restored and the FB is waiting for a confirmation response (*Reset*) from the operator. *Ready* = TRUE *S_GuardLocked* = FALSE *S_UnlockGuard* = FALSE *Error* = FALSE

SF_GuardMonitoring

This safety FB monitors a relevant safety guard and opens or closes the safety guard.

Instruction	Name	FB/FUN	Graphic expression
SF_GuardMonitoring	Safety Guard Monitoring	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_GuardSwitch1	SAFEBOOL	TRUE or FALSE	FALSE	A variable. The input from guard switch 1. FALSE: The guard is open. TRUE: The guard is closed.
S_GuardSwitch2	SAFEBOOL	TRUE or FALSE	FALSE	A variable. The input from guard switch 2. FALSE: The guard is open. TRUE: The guard is closed.
DiscrepancyTime	TIME	Depends on data type.	T#0ms	A constant. It sets the synchronization time to monitor between *S_GuardSwitch1* and *S_GuardSwitch2*.
S_StartReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_AutoReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
Reset	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.

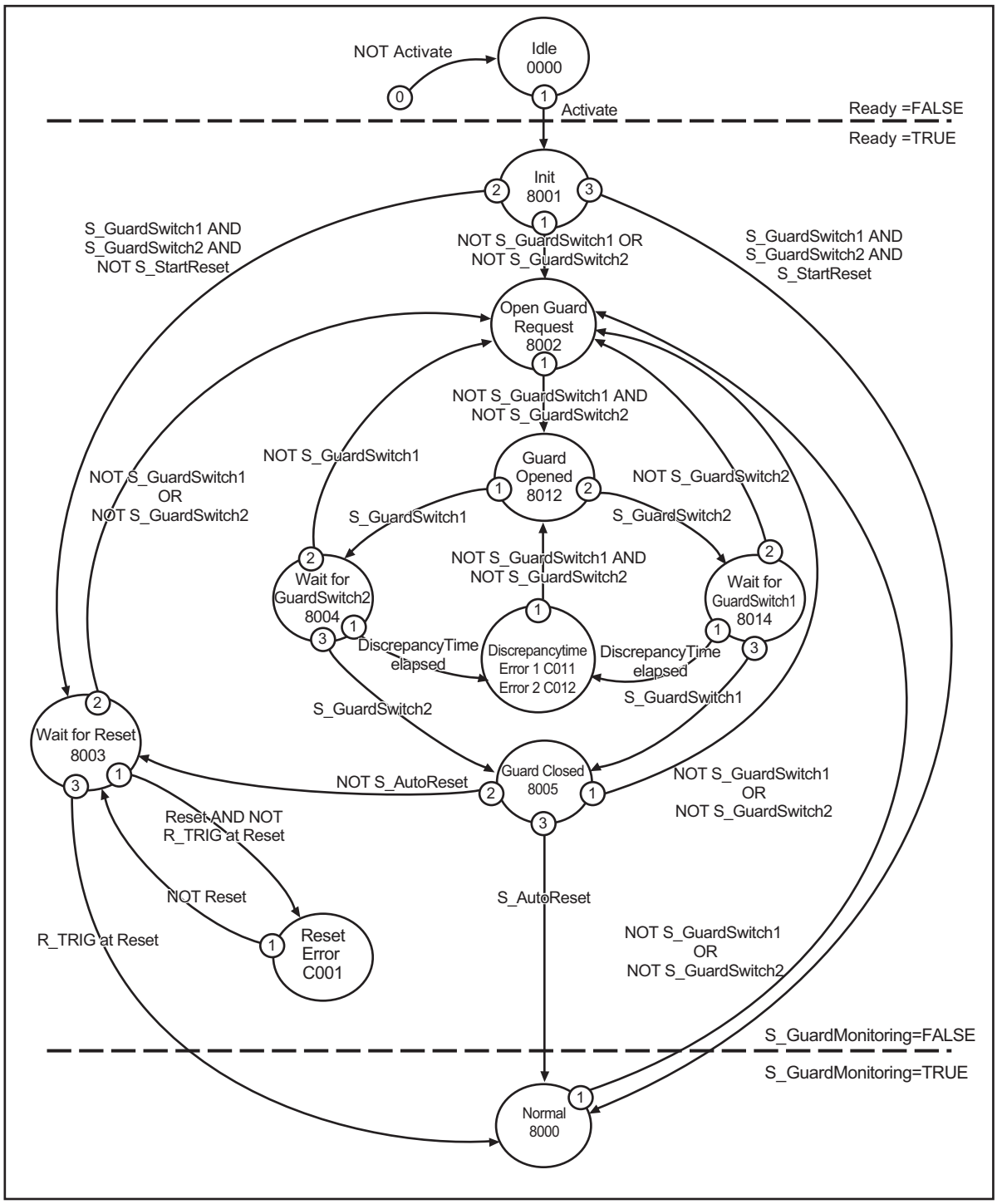
Output Variables

Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_GuardMonitoring	SAFEBOOL	TRUE or FALSE	FALSE	Gives the guard status. FALSE: Opens the guard. TRUE: Closes the guard.
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

Function

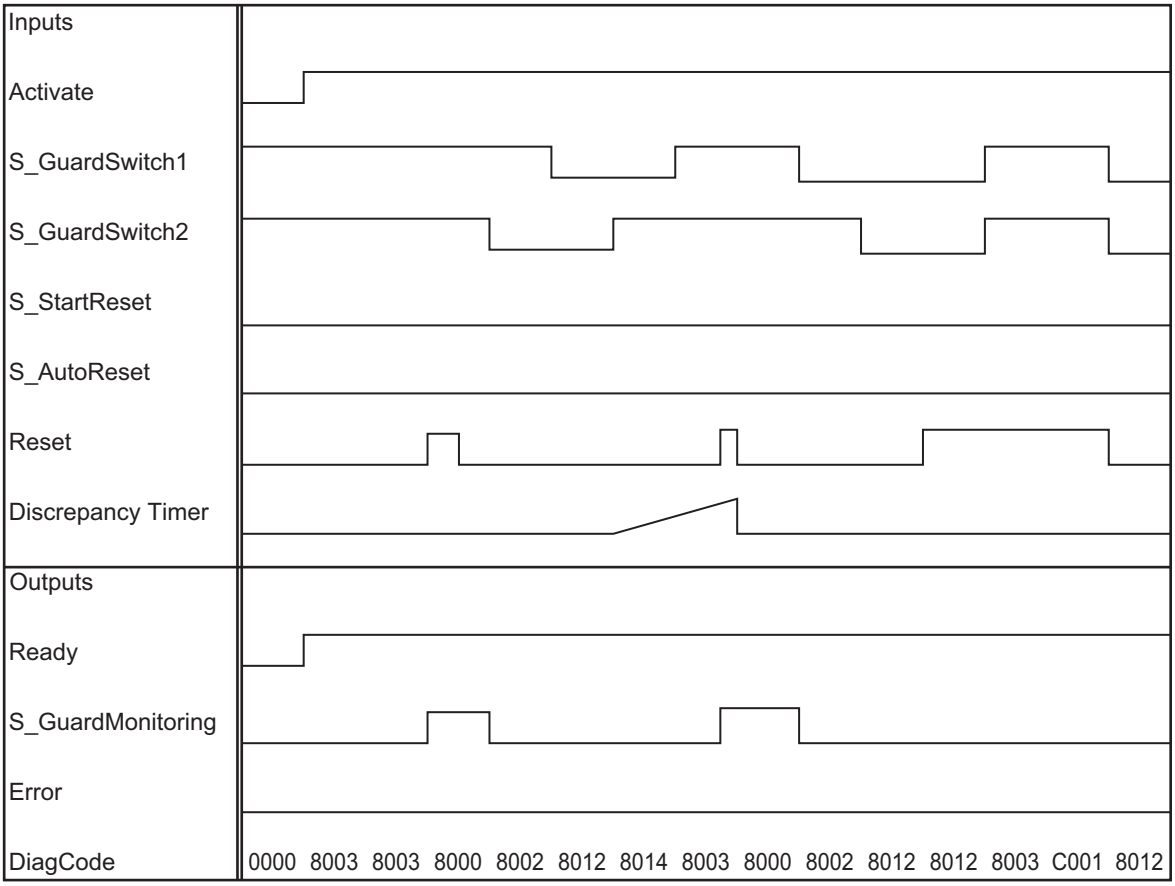
- This FB requires two inputs (*Discrepancy Time* input and *Reset* input) that indicate the guard position of a safety guard with two switches as defined in ISO 14119 (EN 1088). You can bridge the *S_GuardSwitch1* and *S_GuardSwitch2* inputs when there is only one safety guard switch. The monitoring time is the maximum time that is required for both switches to respond when the safety guard is closed. The *Reset, S_StartReset,* and *S_AutoReset* inputs determine how the FB is reset after the safety guard is opened.
- The *S_GuardSwitch1* and *S_GuardSwitch2* inputs must both be changed to FALSE to open the safety guard. The *S_GuardMonitoring* output immediately changes to FALSE when either of the switches is set to FALSE. The *S_GuardSwitch1* and *S_GuardSwitch2* inputs must both be changed to TRUE to close the safety guard.
- This FB monitors the symmetry of the switching operation of both switches. The *S_GuardMonitoring* output remains FALSE if only one of the inputs completes the open or close process.
- The operation of the *S_GuardMonitoring* output depends on the time difference between the switch inputs. If the input values from *S_GuardSwitch1* and *S_GuardSwitch2* are different, monitoring the discrepancy time is started immediately. If the inputs are still different after the discrepancy time expires, the *S_GuardMonitoring* output remains FALSE. If both of the corresponding inputs, *S_GuardSwitch1* and *S_GuardSwitch2*, change to TRUE within the time specified by the *DiscrepancyTime* input, the *S_GuardMonitoring* output is set to TRUE after the confirmation response.
- Activate the *S_StartReset* and *S_AutoReset* inputs only when you can ensure that no hazardous state will occur as the result of starting the Safety CPU Unit.
- A FB error will occur if the same variable is assigned to the input and the discrepancy time is set to 0.
- Set *DiscrepancyTime* to a value that is longer than the safety task period.
  Refer to the *NX-series Safety Control Unit User's Manual* (Cat. No. Z930) for application methods for *DiscrepancyTime*.

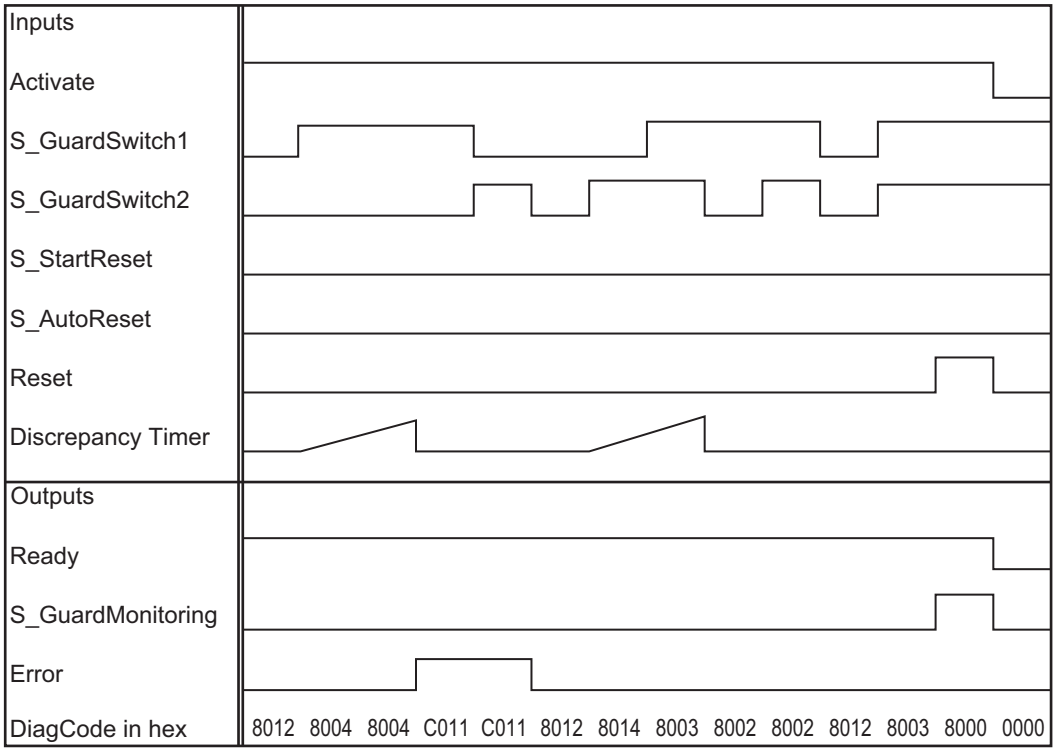
State Transition Diagram



Note Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

Timing Charts





Instruction Execution Errors

Error Detected

- Specific errors are detected for the SAFEBOOL external signal input. According to EN ISO 13849-1 (safety guards that have two switches), mechanical settings and open/close switch settings are combined. According to EN ISO 13849-1, the offset in the response times of both mechanical switches is monitored as the discrepancy time. The offset is treated as detection of an application error, i.e., an error created by the application.
- An error is detected when the offset between the first input and second input for *S_GuardSwitch1* or *S_GuardSwitch2* is larger than the value of the *DiscrepancyTime* input. The *Error* output is set to TRUE.
- If *Reset* is already TRUE when the Wait for Reset state is entered, this FB detects the undetected change to TRUE as an error.

Reset Operation for Errors

- When an error occurs, the *S_GuardMonitoring* output is set to FALSE.
- If the two inputs *S_GuardSwitch1* and *S_GuardSwitch2* are bridged, an error is not detected. To reset the Reset Error state, you must set the *Reset* input to FALSE.
- The *S_GuardSwitch1* and *S_GuardSwitch2* inputs must both be set to FALSE to reset a discrepancy time error.

FB-specific Error Codes

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C001	49153	Reset Error	When the Wait for Reset state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_GuardMonitoring* = FALSE *Error* = TRUE
C011	49169	Discrepancytime Error 1	The discrepancy time expired in state 8004. *Ready* = TRUE *S_GuardMonitoring* = FALSE *Error* = TRUE
C012	49170	Discrepancytime Error 2	The discrepancy time expired in state 8014. *Ready* = TRUE *S_GuardMonitoring* = FALSE *Error* = TRUE

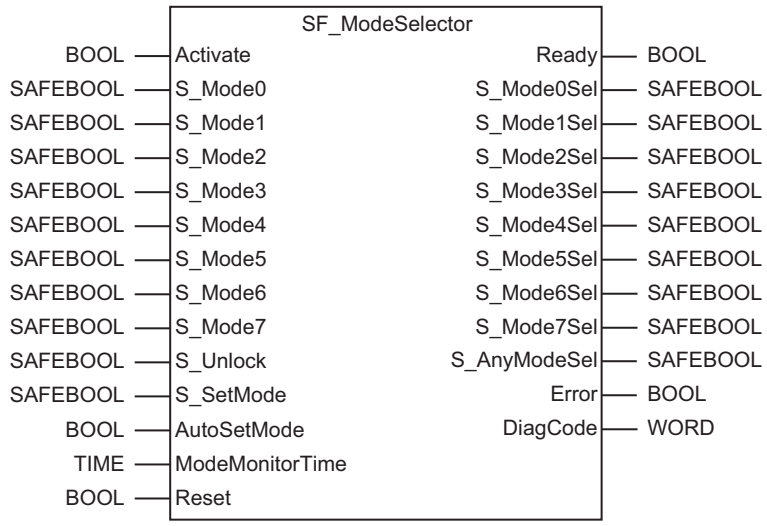
FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *S_GuardMonitoring* = FALSE *Error* = FALSE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8000	32768	Normal	The safety guard is closed and a confirmation response for a safe state was received. *Ready* = TRUE *S_GuardMonitoring* = TRUE *Error* = FALSE
8001	32769	Init	The FB was activated. *Ready* = TRUE *S_GuardMonitoring* = FALSE *Error* = FALSE
8002	32770	Open Guard Request	The FB is waiting for *S_GuardSwitch1* and *S_GuardSwitch2* to change to FALSE. *Ready* = TRUE *S_GuardMonitoring* = FALSE *Error* = FALSE
8003	32771	Wait for Reset	The FB is waiting for *Reset* to change to TRUE. *Ready* = TRUE *S_GuardMonitoring* = FALSE *Error* = FALSE
8012	32786	Guard Opened	The guard is completely open. *Ready* = TRUE *S_GuardMonitoring* = FALSE *Error* = FALSE
8004	32772	Wait for GuardSwitch2	*S_GuardSwitch1* changed to TRUE (waiting for *S_GuardSwitch2*). The discrepancy timer started. *Ready* = TRUE *S_GuardMonitoring* = FALSE *Error* = FALSE
8014	32788	Wait for GuardSwitch1	*S_GuardSwitch2* changed to TRUE (waiting for *S_GuardSwitch1).* The discrepancy timer started. *Ready* = TRUE *S_GuardMonitoring* = FALSE *Error* = FALSE
8005	32773	Guard Closed	The guard was closed. If *S_AutoReset* is FALSE, the FB is waiting for *Reset*. *Ready* = TRUE *S_GuardMonitoring* = FALSE *Error* = FALSE

SF_ModeSelector

This safety FB selects the system operation mode (automatic, manual, semi-automatic, etc.).

Instruction	Name	FB/FUN	Graphic expression
SF_ModeSelector	Mode Selector	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_Mode0	SAFEBOOL	TRUE or FALSE	FALSE	A constant or a variable. It is input 0 from the mode selection switch. FALSE: No request was made by the operator to select mode 0. TRUE: A request was made by the operator to select mode 0.
S_Mode1	SAFEBOOL	TRUE or FALSE	FALSE	A constant or a variable. It is input 1 from the mode selection switch. FALSE: No request was made by the operator to select mode 1. TRUE: A request was made by the operator to select mode 1.
S_Mode2	SAFEBOOL	TRUE or FALSE	FALSE	A constant or a variable. It is input 2 from the mode selection switch. FALSE: No request was made by the operator to select mode 2. TRUE: A request was made by the operator to select mode 2.
S_Mode3	SAFEBOOL	TRUE or FALSE	FALSE	A constant or a variable. It is input 3 from the mode selection switch. FALSE: No request was made by the operator to select mode 3. TRUE: A request was made by the operator to select mode 3.

Variable	Data type	Valid range	Default	Description
S_Mode4	SAFEBOOL	TRUE or FALSE	FALSE	A constant or a variable. It is input 4 from the mode selection switch. FALSE: No request was made by the operator to select mode 4. TRUE: A request was made by the operator to select mode 4.
S_Mode5	SAFEBOOL	TRUE or FALSE	FALSE	A constant or a variable. It is input 5 from the mode selection switch. FALSE: No request was made by the operator to select mode 5. TRUE: A request was made by the operator to select mode 5.
S_Mode6	SAFEBOOL	TRUE or FALSE	FALSE	A constant or a variable. It is input 6 from the mode selection switch. FALSE: No request was made by the operator to select mode 6. TRUE: A request was made by the operator to select mode 6.
S_Mode7	SAFEBOOL	TRUE or FALSE	FALSE	A constant or a variable. It is input 7 from the mode selection switch. FALSE: No request was made by the operator to select mode 7. TRUE: A request was made by the operator to select mode 7.
S_Unlock	SAFEBOOL	TRUE or FALSE	FALSE	A constant or a variable. It locks or unlocks the selected mode. FALSE: The actual *S_ModeXSel* output is locked, so the *S_ModeXSel* output will not change even if the *S_ModeX* input has changed when *S_SetMode* changes to TRUE. TRUE: The mode can be changed because the selected *S_ModeXSel* output is not locked.
S_SetMode	SAFEBOOL	TRUE or FALSE	FALSE	A constant or a variable. If *AutoSetMode* is FALSE, this variable executes the selected mode change. TRUE: The selected mode is set when this variable changes to TRUE. FALSE: If *AutoSetMode* is TRUE, a constant FALSE is set. When the operator changes the mode selection switch, *S_AnyModeSel* and *S_ModeXSel* change to FALSE. The mode change is executed when *S_ModeXSel* changes to TRUE as the result of *S_SetMode* changing to TRUE.
AutoSetMode	BOOL	TRUE or FALSE	FALSE	A constant. It sets mode confirmation. FALSE: To change the mode, the operator must confirm the change with the *S_SetMode* input. TRUE: Even if the operator does not confirm the change with the *S_SetMode*, the *S_ModeXSel* output will change automatically when the *S_ModeX* input changes. (However, the operation is the same as for FALSE if the mode is locked with *S_Unlock*.)
ModeMonitorTime	TIME	Depends on data type.	T#0ms	A constant. It is the maximum allowable time to change the selection input.
Reset	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.

Note “X” in “S_ModeX” and “S_ModeXSel” indicates a number between 0 and 7.

Output Variables

Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_Mode0Sel	SAFEBOOL	TRUE or FALSE	FALSE	Indicates that mode 0 was selected and acknowledged. FALSE: Mode 0 was not selected or the selection was not confirmed. TRUE: Mode 0 was selected and confirmed.
S_Mode1Sel	SAFEBOOL	TRUE or FALSE	FALSE	Indicates that mode 1 was selected and acknowledged. FALSE: Mode 1 was not selected or the selection was not confirmed. TRUE: Mode 1 was selected and confirmed.
S_Mode2Sel	SAFEBOOL	TRUE or FALSE	FALSE	Indicates that mode 2 was selected and acknowledged. FALSE: Mode 2 was not selected or the selection was not confirmed. TRUE: Mode 2 was selected and confirmed.
S_Mode3Sel	SAFEBOOL	TRUE or FALSE	FALSE	Indicates that mode 3 was selected and acknowledged. FALSE: Mode 3 was not selected or the selection was not confirmed. TRUE: Mode 3 was selected and confirmed.
S_Mode4Sel	SAFEBOOL	TRUE or FALSE	FALSE	Indicates that mode 4 was selected and acknowledged. FALSE: Mode 4 was not selected or the selection was not confirmed. TRUE: Mode 4 was selected and confirmed.
S_Mode5Sel	SAFEBOOL	TRUE or FALSE	FALSE	Indicates that mode 5 was selected and acknowledged. FALSE: Mode 5 was not selected or the selection was not confirmed. TRUE: Mode 5 was selected and confirmed.
S_Mode6Sel	SAFEBOOL	TRUE or FALSE	FALSE	Indicates that mode 6 was selected and acknowledged. FALSE: Mode 6 was not selected or the selection was not confirmed. TRUE: Mode 6 was selected and confirmed.
S_Mode7Sel	SAFEBOOL	TRUE or FALSE	FALSE	Indicates that mode 7 was selected and acknowledged. FALSE: Mode 7 was not selected or the selection was not confirmed. TRUE: Mode 7 was selected and confirmed.
S_AnyModeSel	SAFEBOOL	TRUE or FALSE	FALSE	Indicates that one of the modes was selected and acknowledged. FALSE: None of the modes is selected. TRUE: One of the eight modes was selected and confirmed.
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

Note “X” in “S_ModeX” and “S_ModeXSel” indicates a number between 0 and 7.

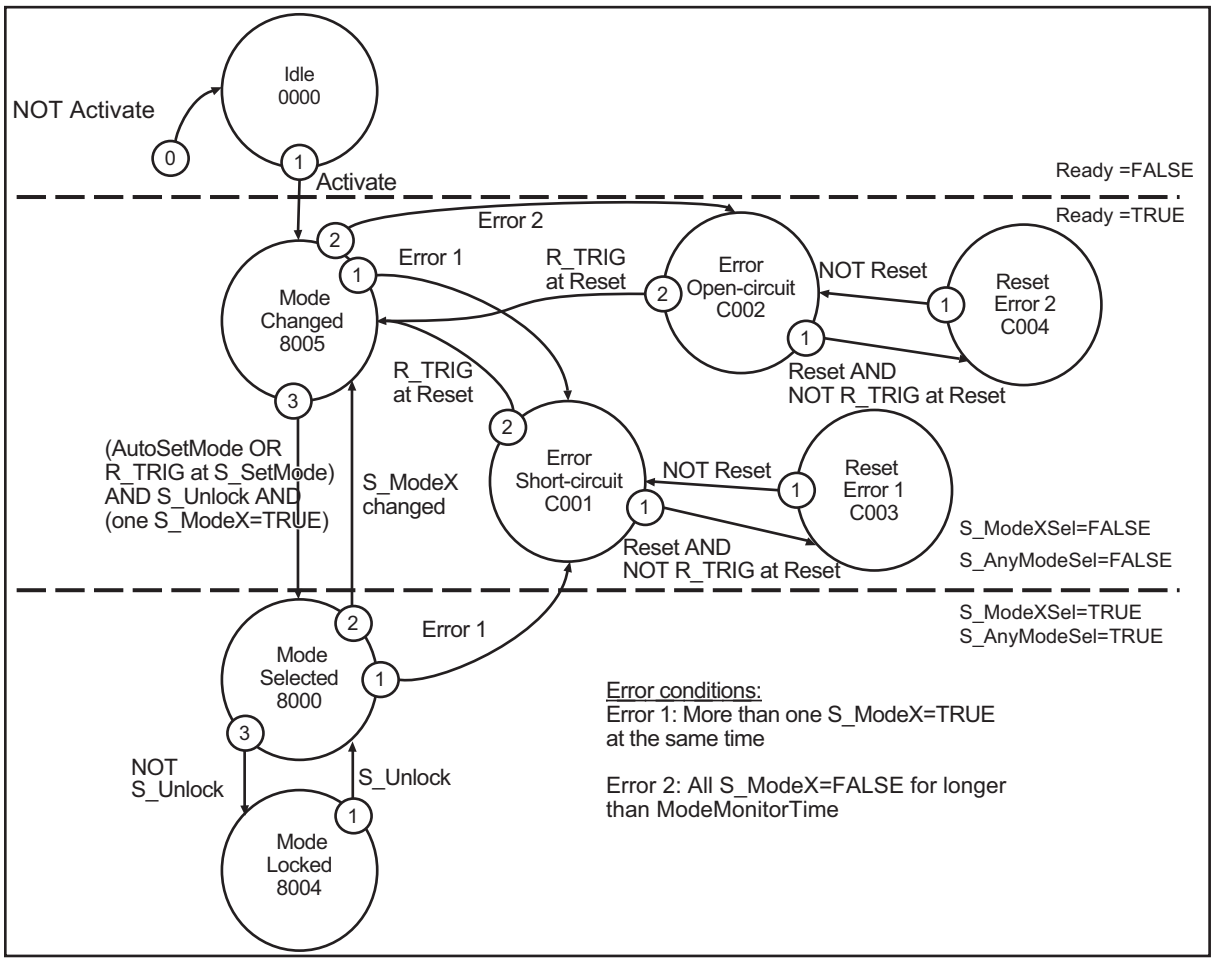
Function

- The FB handles a mode selection switch with up to eight positions.
- You can change the mode to change the operation mode of the machine (automatic mode, manual mode, semi-automatic mode, etc.).
- It is assumed that the machine is in safe mode when the Controller is started. The inputs to the FB (e.g., a machine start button) change the mode to the mode that is set on the mode selection switch when the machine is started.
- The default state after the FB is activated is the ModeChanged state. This is the safe state for the FB, in which all *S_ModeXSel* outputs and the *S_AnyModeSel* output are FALSE.
- When the mode selection switch is changed and the FB is in the ModeChanged state:
  - If *AutoSetMode* is FALSE: A new *S_ModeX* input is acknowledged by *S_SetMode* changing to TRUE to produce a new *S_ModeXSel* output.
  - If *AutoSetMode* is TRUE: A new *S_ModeX* automatically produces a new *S_ModeXSel* output.
  - When an *S_ModeX* input is TRUE, only transitions such as from 8005 to 8000 are valid. As long as *S_ModeX* is FALSE, the FB will retain state 8005 even if a change to TRUE is detected for *S_SetMode*.
- The time is not monitored for a transition from the ModeChanged to the ModeSelected state, i.e., when *S_SetMode* is changed to TRUE by the operator.
- If the FB is in the ModeSelected state when a new *S_ModeX* input (high priority) and an *S_Unlock* signal (low priority) change to FALSE at the same time, the FB will enter the ModeChanged state.
- You can set constants with the default FALSE state for *S_ModeX* inputs that are not used for mode selection to simplify program validation.
- Set the *AutoSetMode* input to TRUE only when you can ensure that no hazardous state will occur as the result of starting the Safety CPU Unit.

Precautions for Correct Use

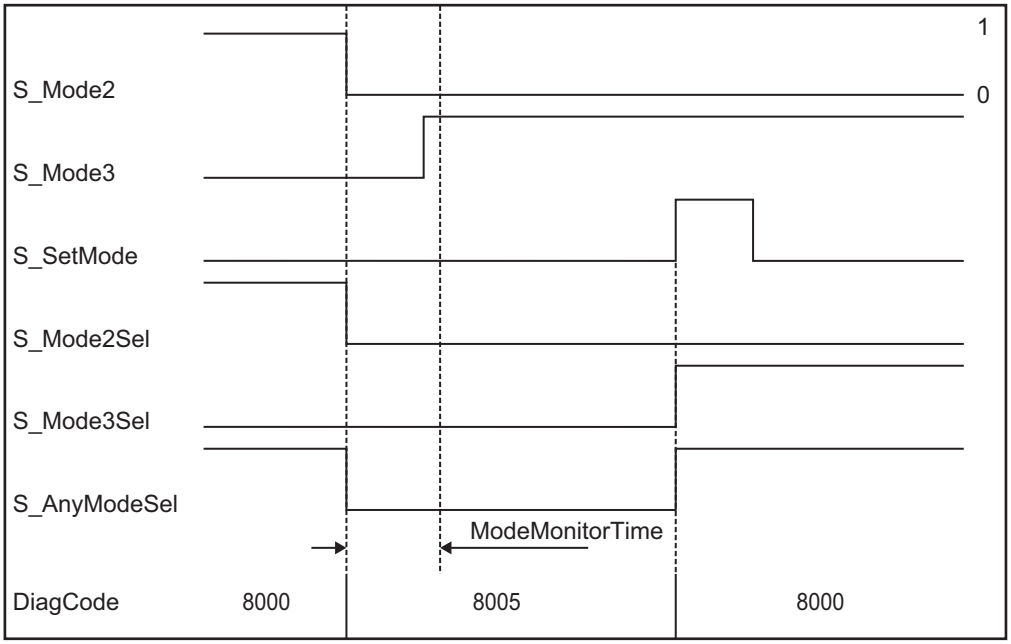
An error will occur for this function if more than one *S_ModeX* input changes to TRUE at the same time. Use a non-shorting mode selection switch.

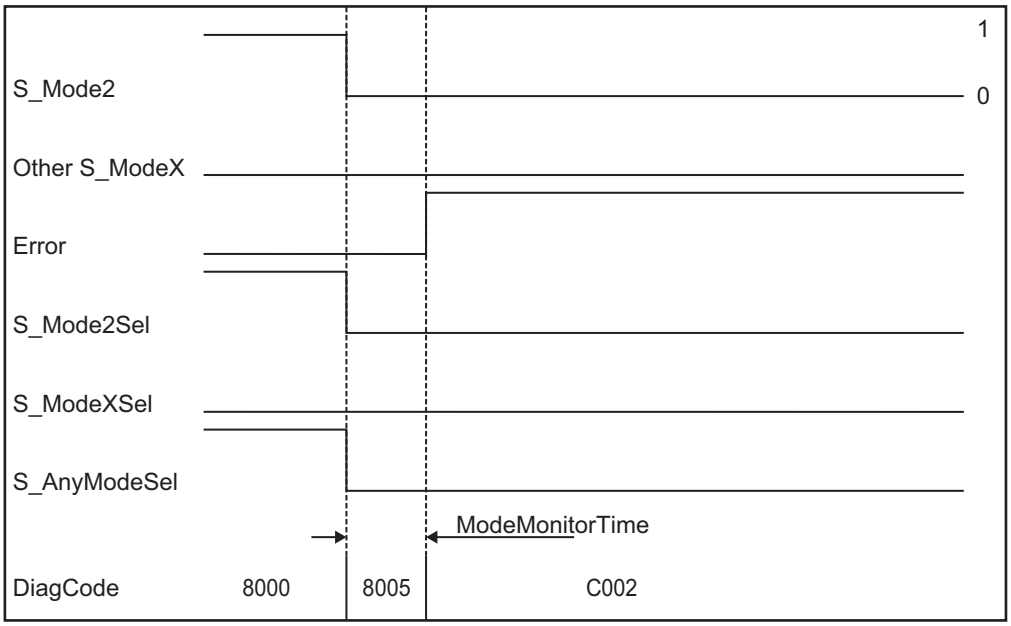
State Transition Diagram

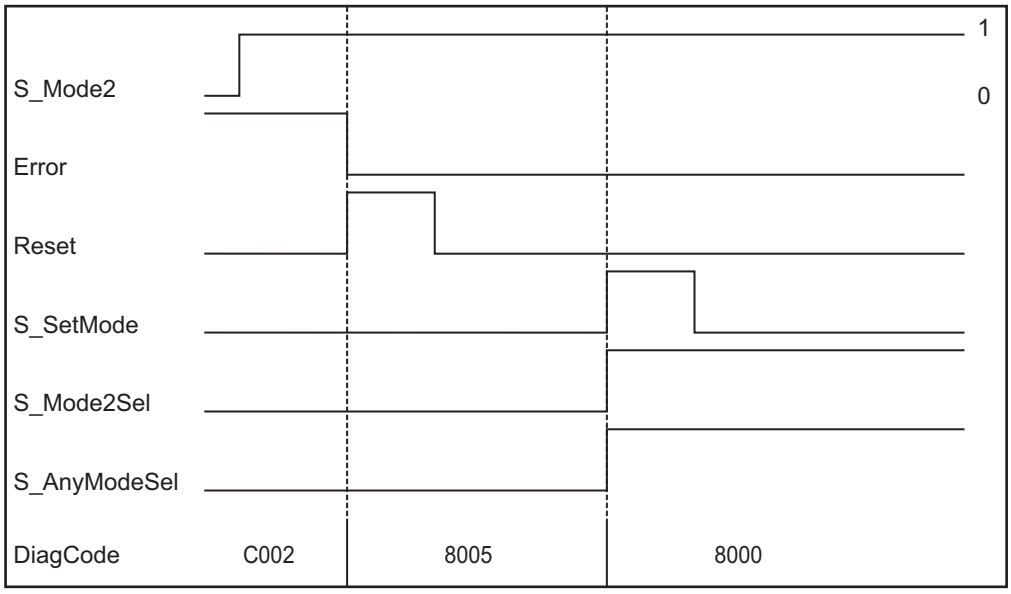


Note Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

Timing Charts






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

Instruction Execution Errors

Error Detected

- The FB detects the absence of a mode selection as an error. This invalid state is detected after *ModeMonitorTime* elapses.
  - The invalid state restarts with each change to FALSE of an *S_ModeX* switched mode input, and the state changes to ModeChanged (8005) after the FB is activated.
- If more than one mode input is selected at the same time, an error is detected immediately.
- If *Reset* is already TRUE when error state C001 or C002 is entered, the FB detects the undetected change to TRUE as an error.

Operation for Errors

- When an error occurs, the *S_ModeXSel* and *S_AnyModeSel* outputs change to their safe states, FALSE. The *DiagCode* output gives the relevant error code and the *Error* output changes to TRUE.
- Errors are recognized when the *Reset* input changes to TRUE and the FB changes from an error state to the ModeChanged state.

FB-specific Error Codes

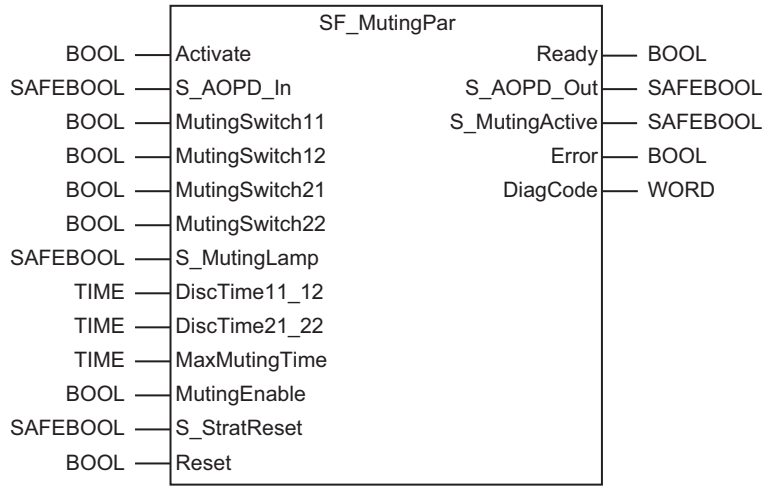
DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C001	49153	Error Short-circuit	The FB detected that more than one *S_ModeX* input is TRUE. For example, the cable may be short-circuited. *Ready* = TRUE *Error* = TRUE *S_AnyModeSel* = FALSE All *S_ModeXSel* = FALSE
C002	49154	Error Open-circuit	The FB detected that all *S_ModeX* inputs are FALSE. The time after an *S_ModeX* input changed exceeded *ModeMonitorTime*. For example, the cable may be broken. *Ready* = TRUE *Error* = TRUE *S_AnyModeSel* = FALSE All *S_ModeXSel* = FALSE
C003	49155	Reset Error 1	When the Error Short-circuit state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *Error* = TRUE *S_AnyModeSel* = FALSE All *S_ModeXSel* = FALSE
C004	49156	Reset Error 2	When the Error Open-circuit state was entered, an undetected change to TRUE in the Reset input was detected. *Ready* = TRUE *Error* = TRUE *S_AnyModeSel* = FALSE All *S_ModeXSel* = FALSE

● FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *Error* = FALSE *S_AnyModeSel* = FALSE All *S_ModeXSel* = FALSE
8005	32773	ModeChanged	The state after the FB is activated, after the *S_ModeX* inputs changed but the mode is not locked, or after an error state was reset. *Ready* = TRUE *Error* = FALSE *S_AnyModeSel* = FALSE All *S_ModeXSel* = FALSE
8000	32768	ModeSelected	A valid mode is selected but not yet locked. *Ready* = TRUE *Error* = FALSE *S_AnyModeSel* = TRUE All *S_ModeXSel* = Selected X is TRUE and others are FALSE.
8004	32772	ModeLocked	A valid mode is selected and locked. *Ready* = TRUE *Error* = FALSE *S_AnyModeSel* = TRUE All *S_ModeXSel* = Selected X is TRUE and others are FALSE.

SF_MutingPar

Muting is used to intentionally disable a safety function. This safety FB performs parallel muting with four muting sensors.

Instruction	Name	FB/FUN	Graphic expression
SF_MutingPar	Parallel Muting	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_AOPD_In	SAFEBOOL	TRUE or FALSE	FALSE	This is the OSSD (safety output) signal from the AOPD (active optoelectronic protective device). FALSE: Something entered the protected area. TRUE: Nothing entered the protected area.
MutingSwitch11	BOOL*1	TRUE or FALSE	FALSE	A variable. It is the status of muting sensor 11. FALSE: Muting sensor 11 is not operating. TRUE: A workpiece activated muting sensor 11.
MutingSwitch12	BOOL*1	TRUE or FALSE	FALSE	A variable. It is the status of muting sensor 12. FALSE: Muting sensor 12 is not operating. TRUE: A workpiece activated muting sensor 12.
MutingSwitch21	BOOL*1	TRUE or FALSE	FALSE	A variable. It is the status of muting sensor 21. FALSE: Muting sensor 21 is not operating. TRUE: A workpiece activated muting sensor 21.
MutingSwitch22	BOOL*1	TRUE or FALSE	FALSE	A variable. It is the status of muting sensor 22. FALSE: Muting sensor 22 is not operating. TRUE: A workpiece activated muting sensor 22.
S_MutingLamp	SAFEBOOL	TRUE or FALSE	FALSE	A constant or a variable. It is the muting lamp status input (e.g., filament broken status). FALSE: Muting lamp failure. TRUE: Muting lamp is normal.

Variable	Data type	Valid range	Default	Description
DiscTime11_12	TIME	T#0s to T#4s	T#0s	A constant. It sets the maximum discrepancy time between *MutingSwitch11* and *MutingSwitch12*.
DiscTime21_22	TIME	T#0s to T#4s	T#0s	A constant. It sets the maximum discrepancy time between *MutingSwitch21* and *MutingSwitch22*.
MaxMutingTime	TIME	T#0s to T#10min	T#0s	A constant. It sets the maximum time until completion of the muting sequence. The timer starts when the muting sensor first operates.
MutingEnable	BOOL	TRUE or FALSE	FALSE	A constant or a variable. It is a command from the control system to enable starting muting as required in the machine cycle. You can change this signal to OFF after muting starts. FALSE: Disables muting. TRUE: Enables starting muting.
S_StartReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
Reset	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.

*1. You must connect a SAFEBOOL variable (not a BOOL variable) depending on safety requirements.

Output Variables

Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_AOPD_Out	SAFEBOOL	TRUE or FALSE	FALSE	This safety-related output gives the status of the protection devices that is being muted. FALSE: Something has entered the AOPD protected area and muting is disabled. TRUE: Nothing has entered the AOPD protected area and muting is enabled.
S_MutingActive	SAFEBOOL	TRUE or FALSE	FALSE	Gives the muting status. FALSE: Muting is disabled. TRUE: Muting is enabled.
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

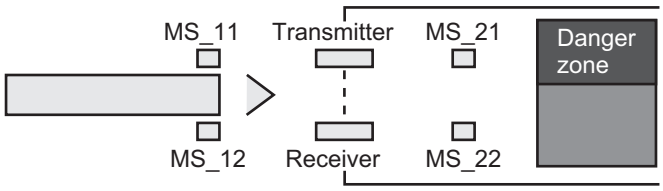
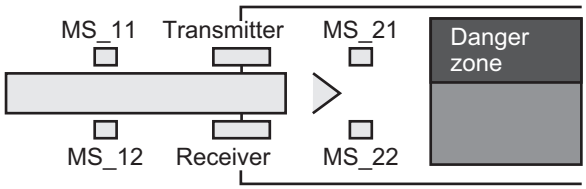
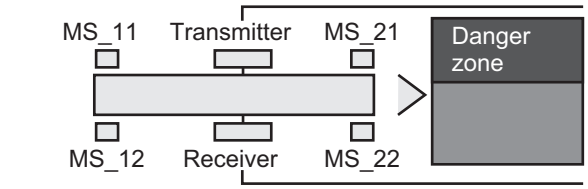
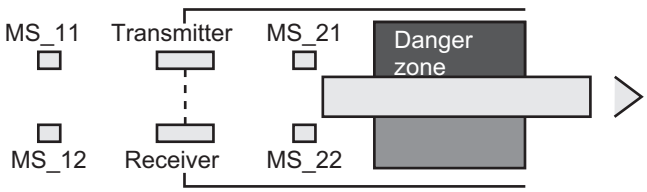
Precautions for Correct Use

This FB does not detect short-circuits in muting sensor signals or errors in the function applications that supply those signals. It interprets them as illegal muting sequences. Unintentional muting must not be allowed under these conditions. Give attention to this during risk assessment.

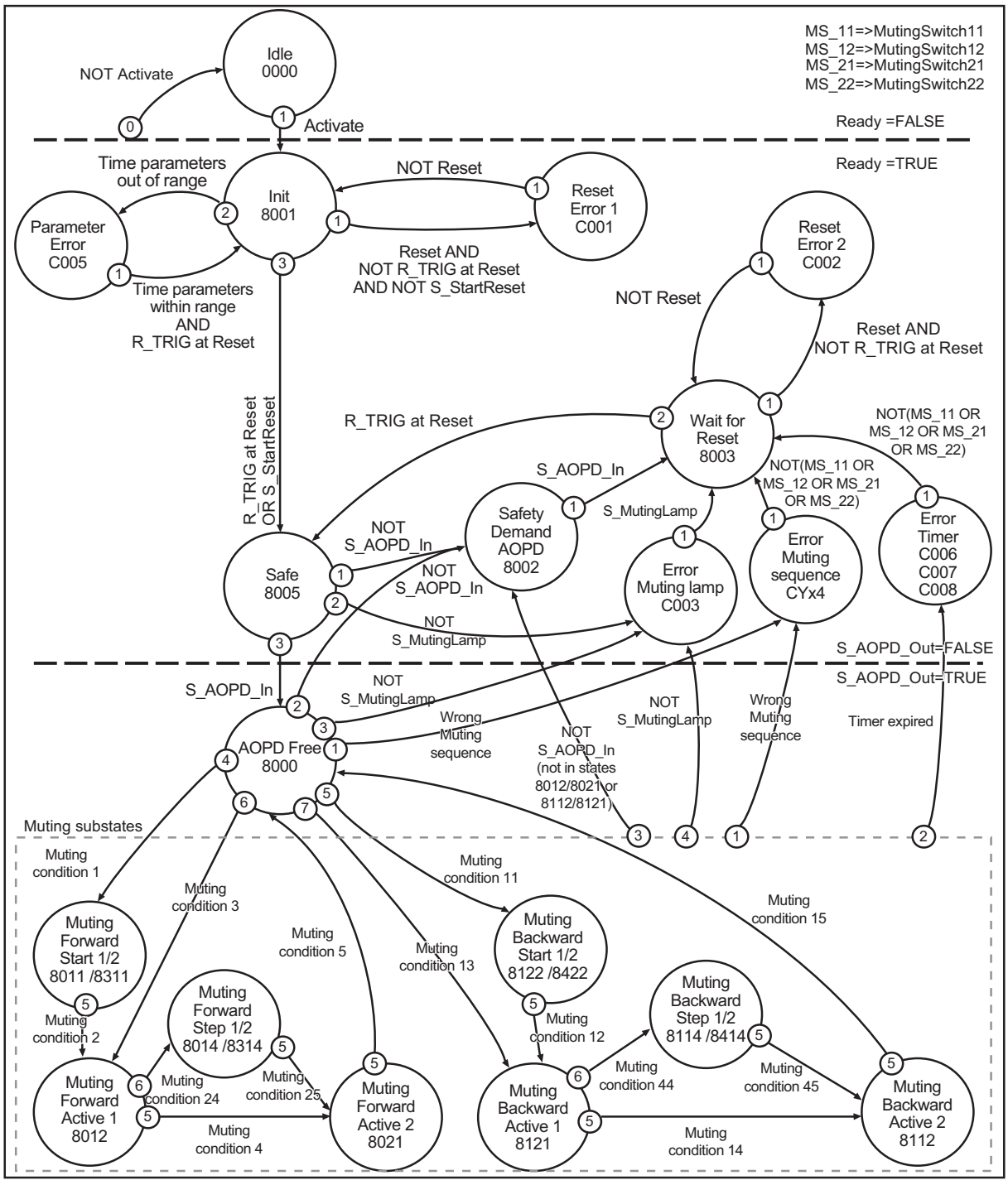
Function

- Muting is used to intentionally disable a safety function. Muting is used, for example, to pass a workpiece through a hazardous area without stopping the machine. Muting is activated by muting sensors. Four muting sensors are used. To correctly incorporate the safety function into a manufacturing process, you must ensure that people will not enter the hazardous area while the light curtain is being muted. Proximity sensors, photoelectric barriers, limit switches, and other devices that do not have failsafe mechanisms are used for muting sensors. Muting operation must be indicated by indicator lights.
- There are two types of muting: parallel and sequential. This FB performs parallel muting with four muting sensors. Passing into a hazardous area in the forward direction is described below. (Refer to *SF_MutingPar Instruction Application Example for Forward Entry with Four Sensors* on page 4-72.) The FB can be used for either forward or backward passage. To prevent manual operation, muting must also be enabled by process control with the *MutingEnable* signal to perform muting.
- The input parameters to the FB include four muting sensor signals (*MutingSwitch11* to *MutingSwitch22*), an OSSD signal from a photoelectric protection device (*S_AOPD_In*), and three time parameters (*DiscTime11_12*, *DiscTime21_22*, and *MaxMutingTime*).
- Activate the *S_StartReset* input only when you can ensure that no hazardous state will occur as the result of starting the Safety CPU Unit.
- A FB error will occur if the same variable is assigned to the input and the discrepancy time is set to 0.

● SF_MutingPar Instruction Application Example for Forward Entry with Four Sensors

Order	Diagram	Description
1		Muting mode is enabled (*S_MutingActive* = TRUE) when the passage of a workpiece turns ON *MutingSwitch11* (MS_11) and *MutingSwitch12* (MS_12) within the time set by *DiscTime11_12*.
2		Muting mode is enabled as long as *MutingSwitch11* (MS_11) and *MutingSwitch12* (MS_12) remain ON due to the workpiece. This allows the workpiece to pass through the light curtain without stopping the machine.
3		*MutingSwitch21* (MS_21) and *MutingSwitch22* (MS_22) must turn ON before *MutingSwitch11* (MS_11) and *MutingSwitch12* (MS_12) turn OFF. This ensures that muting mode remains ON. The discrepancy time between *MutingSwitch21* and *MutingSwitch22* is monitored for the time that is set in *DiscTime21_22*.
4		Muting mode is ended if only *MutingSwitch22* (MS_22) is ON. Muting mode is ended if *MutingSwitch21* (MS_21) or *MutingSwitch22* (MS_22) turns OFF due to the workpiece. The maximum time that the muting mode is enabled is set by *MaxMutingTime*.

State Transition Diagram



Note 1. Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

2. A state transition to Error Muting sequence (priority 1), Error Timer (priority 2), Safety Demand AOPD (priority 3), or Error Muting lamp (priority 4) in the muting substates has higher priority than a state transition to muting substates with priority 5 or 6.

3. The muting conditions are described below.

Forward Direction

Muting Condition 1 (8000 to 8011) (when MS_11 is the first switch to start muting)
Timers for *MaxMutingTime* and *DiscTime11_12* started: *MutingEnable* AND (R_TRIG at MS_11 AND NOT MS_12 AND NOT MS_21 AND NOT MS_22)

Muting Condition 1 (8000 to 8311) (when MS_12 is the first switch to start muting)
Timers for *MaxMutingTime* and *DiscTime11_12* started: *MutingEnable* AND (NOT MS_11 AND R_TRIG at MS_12 AND NOT MS_21 AND NOT MS_22)

Muting Condition 2 (8011 to 8012) (when MS_12 is the second switch to start muting)
Timer for *DiscTime11_12* stopped: *MutingEnable* AND (MS_11 AND R_TRIG at MS_12 AND NOT MS_21 AND NOT MS_22)

Muting Condition 2 (8311 to 8012) (when MS_11 is the second switch to start muting)
Timer for *DiscTime11_12* stopped: *MutingEnable* AND (R_TRIG at MS_11 AND MS_12 AND NOT MS_21 AND NOT MS_22)

Muting Condition 3 (8000 to 8012) (when both switches simultaneously start muting)
Timer for *MaxMutingTime* started: *MutingEnable* AND (R_TRIG at MS_11 AND R_TRIG at MS_12 AND NOT MS_21 AND NOT MS_22)

Muting Condition 4 (8012 to 8021) (when all switches are active)
MS_11 AND MS_12 AND MS_21 AND MS_22

Muting Condition 24 (8012 to 8014) (when MS_21 is the first switch to stop muting)
Timer for *DiscTime21_22* started: MS_11 AND MS_12 AND R_TRIG at MS_21 AND NOT MS_22

Muting Condition 24 (8012 to 8314) (when MS_22 is the first switch to stop muting)
Timer for *DiscTime21_22* started: MS_11 AND MS_12 AND NOT MS_21 AND R_TRIG at MS_22

Muting Condition 25 (8014 to 8021) (when MS_22 is the second switch to stop muting)
Timer for *DiscTime21_22* stopped: MS_11 AND MS_12 AND MS_21 AND R_TRIG at MS_22

Muting Condition 25 (8314 to 8021) (when MS_21 is the second switch to stop muting)
Timer for *DiscTime21_22* stopped: MS_11 AND MS_12 AND R_TRIG at MS_21 AND MS_22

Muting Condition 5 (8021 to 8000) (when one of the muting stop switches is reset)
Timer for *MaxMutingTime* stopped: NOT MS_11 AND NOT MS_12 AND (F_TRIG at MS_21 OR F_TRIG at MS_22)

Backward Direction

Muting Condition 11 (8000 to 8122) (when MS_21 is the first switch to start muting)
Timers for *MaxMutingTime* and *DiscTime21_22* started: *MutingEnable* AND (NOT MS_22 AND R_TRIG at MS_21 AND NOT MS_11 AND NOT MS_12)

Muting Condition 11 (8000 to 8422) (when MS_22 is the first switch to start muting)
Timers for *MaxMutingTime* and *DiscTime21_22* started: *MutingEnable* AND (R_TRIG at MS_22 AND NOT MS_21 AND NOT MS_11 AND NOT MS_12)

Muting Condition 12 (8122 to 8121) (when MS_22 is the second switch to start muting)
Timer for *DiscTime21_22* stopped: *MutingEnable* AND (MS_21 AND R_TRIG at MS_22 AND NOT MS_11 AND NOT MS_12)

Muting Condition 12 (8422 to 8121) (when MS_21 is the second switch to start muting)
Timer for *DiscTime21_22* stopped: *MutingEnable* AND (R_TRIG at MS_21 AND MS_22 AND NOT MS_11 AND NOT MS_12)

Muting Condition 13 (8000 to 8121) (when both switches simultaneously start muting)
Timer for *MaxMutingTime* started: *MutingEnable* AND (R_TRIG at MS_21 AND R_TRIG at MS_22 AND NOT MS_11 AND NOT MS_12)

Muting Condition 14 (8121 to 8112) (when all switches are active)
MS_11 AND MS_12 AND MS_21 AND MS_22

Muting Condition 44 (8121 to 8114) (when MS_11 is the first switch to stop muting)
Timer for *DiscTime11_12* started: MS_21 AND MS_22 AND R_TRIG at MS_11 AND NOT MS_12

Muting Condition 44 (8121 to 8414) (when MS_12 is the first switch to stop muting)
Timer for *DiscTime11_12* started: MS_21 AND MS_22 AND NOT MS_11 AND R_TRIG at MS_12

Muting Condition 45 (8114 to 8112) (when MS_12 is the second switch to stop muting)
Timer for *DiscTime11_12* stopped: MS_21 AND MS_22 AND MS_11 AND R_TRIG at MS_12

Muting Condition 45 (8414 to 8112) (when MS_11 is the second switch to stop muting)
Timer for *DiscTime11_12* stopped: MS_21 AND MS_22 AND R_TRIG at MS_11 AND MS_12

Muting Condition 15 (8112 to 8000) (when one of the muting stop switches is reset)
Timer for *MaxMutingTime* stopped: NOT MS_21 AND NOT MS_22 AND (F_TRIG at MS_11 OR F_TRIG at MS_12)

Illegal Muting Sequences:

State 8000:

(*MutingEnable* = FALSE at start of muting sequence) OR

((MS_11 OR MS_12) AND (MS_21 OR MS_22)) OR

(R_TRIG at MS_11 AND MS_12 AND NOT R_TRIG at MS_12) OR

(R_TRIG at MS_12 AND MS_11 AND NOT R_TRIG at MS_11) OR

(R_TRIG at MS_21 AND MS_22 AND NOT R_TRIG at MS_22) OR

(R_TRIG at MS_22 AND MS_21 AND NOT R_TRIG at MS_21) OR

((MS_11 AND NOT R_TRIG at MS_11) AND (MS_12 AND NOT R_TRIG at MS_12)) OR

((MS_21 AND NOT R_TRIG at MS_21) AND (MS_22 AND NOT R_TRIG at MS_22))

State 8011: NOT *MutingEnable* OR NOT MS_11 OR MS_21 OR MS_22

State 8311: NOT *MutingEnable* OR NOT MS_12 OR MS_21 OR MS_22

State 8012: NOT MS_11 OR NOT MS_12

State 8021: R_TRIG at MS_11 OR R_TRIG at MS_12 OR R_TRIG at MS_21 OR R_TRIG at MS_22

State 8014: NOT MS_11 OR NOT MS_12 OR NOT MS_21

State 8314: NOT MS_11 OR NOT MS_12 OR NOT MS_22

State 8122: NOT *MutingEnable* OR MS_11 OR MS_12 OR NOT MS_21

State 8422: NOT *MutingEnable* OR MS_11 OR MS_12 OR NOT MS_22

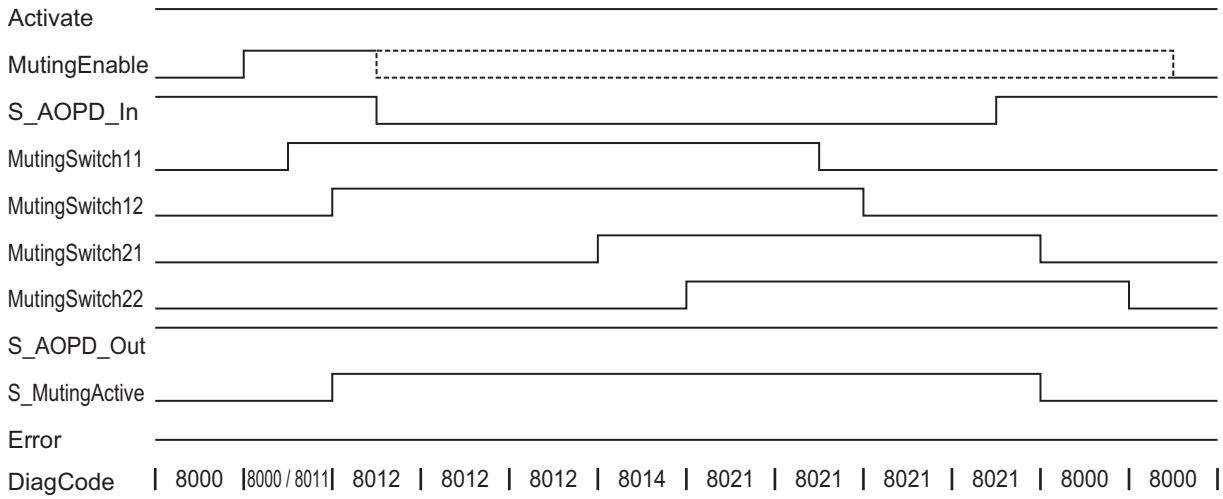
State 8121: NOT MS_21 OR NOT MS_22

State 8112: R_TRIG at MS_11 OR R_TRIG at MS_12 OR R_TRIG at MS_21 OR R_TRIG at MS_22

State 8114: NOT MS_21 OR NOT MS_22 OR NOT MS_11

State 8414: NOT MS_21 OR NOT MS_22 OR NOT MS_12

Timing Charts



Instruction Execution Errors

Error Detected

The FB detects the following errors.

- *DiscTime11_12* or *DiscTime21_22* is set to less than T#0s or more than T#4s.
- *MaxMutingTime* is set to less than T#0s or more than T#10min.
- The discrepancy time for a sensor pair (*MutingSwitch11* and *MutingSwitch12*, or *MutingSwitch21* and *MutingSwitch22*) exceeded the set value.
- Muting (*S_MutingActive* = TRUE) exceeded *MaxMutingTime* (maximum muting time).
- The muting sensors (*MutingSwitch11*, *MutingSwitch12*, *MutingSwitch21*, and *MutingSwitch22*) operated in an incorrect order.
- A muting sequence started without being enabled by *MutingEnable*.
- A muting lamp failure was indicated (*S_MutingLamp* = FALSE).
- An undetected change to TRUE in the *Reset* input was detected in state 8001 or 8003.

Operation for Errors

- When an error occurs, the *S_AOPD_Out* and *S_MutingActive* outputs are set to FALSE. The *DiagCode* output gives the relevant error code and the *Error* output is set to TRUE.
- Operation is not restarted until the error is reset and the operator acknowledges the safe state with *Reset*.

FB-specific Error Codes

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C001	49153	Reset Error 1	When the Init state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C002	49154	Reset Error 2	When the Wait for Reset state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE
C003	49155	Error Muting lamp	An error was detected in the muting lamp. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE
CYx4	--- *1	Error Muting sequence	A muting sequence error was detected in state 8000, 8011, 8311, 8012, 8021, 8014, 8314, 8122, 8422, 8121, 8112, 8114, or 8414. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE Y = Sequence state (forward direction: 6 states, backward direction: 6 states) C0x4 = Error occurred in state 8000. C1x4 = Error occurred in forward direction in state 8011. C2x4 = Error occurred in forward direction in state 8311. C3x4 = Error occurred in forward direction in state 8012. C4x4 = Error occurred in forward direction in state 8014. C5x4 = Error occurred in forward direction in state 8314. C6x4 = Error occurred in forward direction in state 8021. C7x4 = Error occurred in backward direction in state 8122. C8x4 = Error occurred in backward direction in state 8422. C9x4 = Error occurred in backward direction in state 8121. CAx4 = Error occurred in backward direction in state 8114. CBx4 = Error occurred in backward direction in state 8414. CCx4 = Error occurred in backward direction in state 8112. CFx4 = *MutingEnable* was not detected. x = Sensor status when the error occurred (4 bits: LSB = MS_11; MS_12; MS_21; MSB = MS_22).

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C005	49157	Parameter Error	The value of *DiscTime11_12, DiscTime21_22* or *MaxMutingTime* is outside of the valid range. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE
C006	49158	Error Timer MaxMuting	Timing error: The muting operation time while *S_MutingActive* = TRUE exceeded *MaxMuting-Time*. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE
C007	49159	Error Timer MS11_12	Timing error: Discrepancy time between *Muting-Switch11* and *MutingSwitch12* exceeded *Disc-Time11_12.* *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE
C008	49160	Error Timer MS21_22	Timing error: Discrepancy time between *Muting-Switch21* and *MutingSwitch22* exceeded *Disc-Time21_22.* *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE

*1. Find the DiagCode hexadecimal value with the information given in *Status description and output results* and then convert it to the DiagCode decimal value.

● FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = FALSE
8000	32768	AOPD Free	Muting is disabled but the control input from AOPD is active. If the muting timer is operating, it stops. *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = FALSE *Error* = FALSE

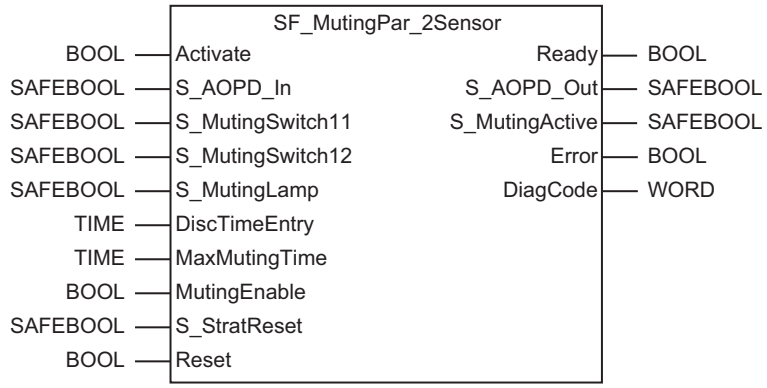
DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8001	32769	Init	The FB was started. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = FALSE
8002	32770	Safety Demand AOPD	Muting is disabled. The control input from AOPD is disabled. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = FALSE
8003	32771	Wait for Reset	A safety request or error was detected and cleared. The operator must respond with *Reset*. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = FALSE
8005	32773	Safe	The safety function is operating. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = FALSE
8011	32785	Muting Forward Start 1	A forward muting sequence is being started after *MutingSwitch11* changed to TRUE. Monitoring is active for *DiscTime11_12.* Monitoring is active for *MaxMutingTime.* *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = FALSE *Error* = FALSE
8311	33553	Muting Forward Start 2	A forward muting sequence is being started after *MutingSwitch12* changed to TRUE. Monitoring is active for *DiscTime11_12.* Monitoring is active for *MaxMutingTime.* *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = FALSE *Error* = FALSE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8012	32786	Muting Forward Active 1	A forward muting sequence is in operation for one of the following. • A change to TRUE was detected in the second muting switch of *MutingSwitch11* and *MutingSwitch12*. • Both *MutingSwitch11* and *MutingSwitch12* were started in the same cycle. Monitoring for *DiscTime11_12* was stopped. Monitoring for *MaxMutingTime* is started after a transition directly from state 8000. *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = TRUE *Error* = FALSE
8014	32788	Muting Forward Step 1	A forward muting sequence is in operation. *MutingSwitch21* operates as the first muting stop switch. Monitoring for *DiscTime21_22* is started. Ready = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = TRUE *Error* = FALSE
8314	33556	Muting Forward Step 2	A forward muting sequence is in operation. *MutingSwitch22* operates as the first muting stop switch. Monitoring for *DiscTime21_22* is started. *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = TRUE *Error* = FALSE
8021	32801	Muting Forward Active 2	A forward muting sequence is still in operation. Both *MutingSwitch21* and *MutingSwitch22* are active, so monitoring for *DiscTime21_22* was stopped. *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = TRUE *Error* = FALSE
8122	33058	Muting Backward Start 1	A backward muting sequence is being started after *MutingSwitch21* changed to TRUE. Monitoring is active for *DiscTime21_22.* Monitoring is active for *MaxMutingTime.* *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = FALSE *Error* = FALSE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8422	33826	Muting Backward Start 2	A backward muting sequence is being started after *MutingSwitch22* changed to TRUE. Monitoring is active for *DiscTime21_22.* Monitoring is active for *MaxMutingTime.* *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = FALSE *Error* = FALSE
8121	33057	Muting Backward Active 1	A backward muting sequence is in operation for one of the following. • A change to TRUE was detected in the second muting switch of *MutingSwitch21* and *MutingSwitch22*. • Both *MutingSwitch21* and *MutingSwitch22* were started in the same cycle. Monitoring for *DiscTime21_22* was stopped. Monitoring for *MaxMutingTime* is started after a transition directly from state 8000. *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = TRUE *Error* = FALSE
8114	33044	Muting Backward Step 1	A backward muting sequence is in operation. *MutingSwitch11* operates as the first muting stop switch. Monitoring for *DiscTime11_12* is started. *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = TRUE *Error* = FALSE
8414	33812	Muting Backward Step 2	A backward muting sequence is in operation. *MutingSwitch12* operates as the first muting stop switch. Monitoring for *DiscTime11_12* is started. *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = TRUE *Error* = FALSE
8112	33042	Muting Backward Active 2	A backward muting sequence is still in operation. Both *MutingSwitch11* and *MutingSwitch12* are active, so monitoring for *DiscTime11_12* was stopped. *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = TRUE *Error* = FALSE

SF_MutingPar_2Sensor

Muting is used to intentionally disable a safety function. This safety FB performs parallel muting with two muting sensors.

Instruction	Name	FB/FUN	Graphic expression
SF_MutingPar_2-Sensor	Parallel Muting with 2 Sensors	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_AOPD_In	SAFEBOOL	TRUE or FALSE	FALSE	A variable. This is the OSSD (safety output) signal from the AOPD (active optoelectronic protective device). FALSE: Something entered the protected area. TRUE: Nothing entered the protected area.
S_Muting-Switch11	SAFEBOOL	TRUE or FALSE	FALSE	A variable. It is the status of muting sensor 11. FALSE: Muting sensor 11 is not operating. TRUE: A workpiece activated muting sensor 11.
S_Muting-Switch12	SAFEBOOL	TRUE or FALSE	FALSE	A variable. It is the status of muting sensor 12. FALSE: Muting sensor 12 is not operating. TRUE: A workpiece activated muting sensor 12.
S_MutingLamp	SAFEBOOL	TRUE or FALSE	FALSE	A constant or a variable. It is the muting lamp status input (e.g., filament broken status). FALSE: Muting lamp failure. TRUE: Muting lamp is normal.
DiscTimeEntry	TIME	T#0s to T#4s	T#0s	A constant. It inputs the maximum discrepancy time between *MutingSwitch11* and *MutingSwitch12*.
MaxMutingTime	TIME	T#0s to T#10min	T#0s	A constant. It inputs the maximum time until completion of the muting sequence. The timer starts when the muting sensor first operates.

Variable	Data type	Valid range	Default	Description
MutingEnable	BOOL	TRUE or FALSE	FALSE	A constant or a variable. It is a command from the control system to enable starting muting as required in the machine cycle. You can change this signal to OFF after muting starts. FALSE: Disables muting. TRUE: Enables starting muting.
S_StartReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
Reset	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.

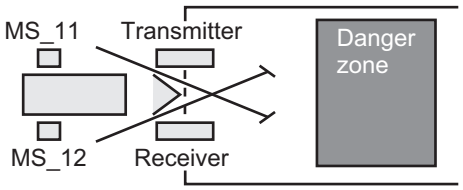
Output Variables

Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4
S_AOPD_Out	SAFEBOOL	TRUE or FALSE	FALSE	This safety output gives the status of the protection devices that is being muted. FALSE: Something has entered the AOPD protected area and muting is disabled. TRUE: Nothing has entered the AOPD protected area and muting is enabled.
S_MutingActive	SAFEBOOL	TRUE or FALSE	FALSE	Gives the muting status. FALSE: Muting is disabled. TRUE: Muting is enabled.
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

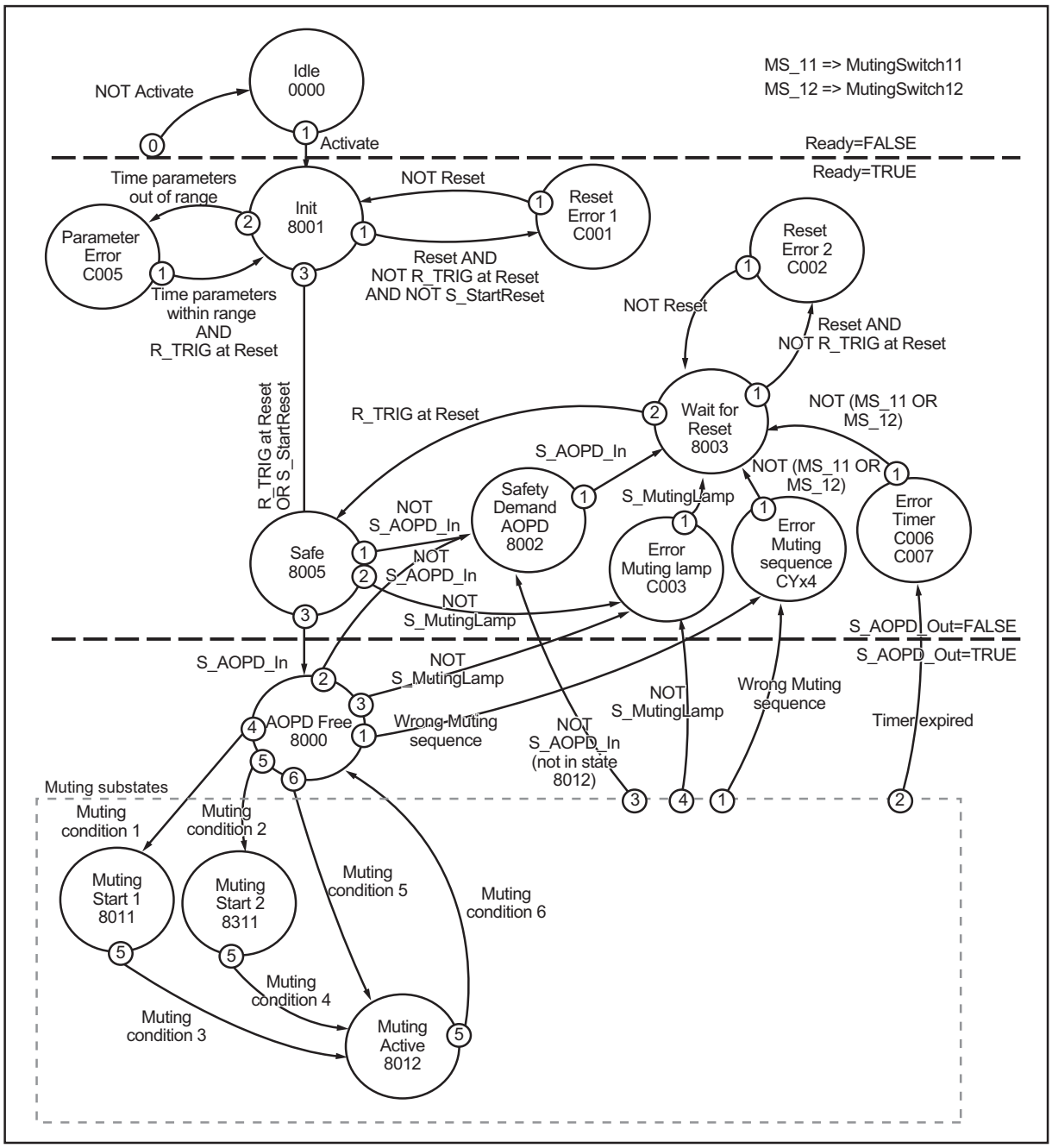
Function

- Muting is used to intentionally disable a safety function. Muting is used, for example, to pass a workpiece through a hazardous area without stopping the machine. Muting is activated by muting sensors. Two muting sensors are used. To correctly incorporate the safety function into a manufacturing process, you must ensure that people will not enter the hazardous area while the light curtain is being muted. Pushbuttons, proximity sensors, photoelectric barriers, limit switches, and other devices that do not have failsafe mechanisms are used for muting sensors. Muting operation must be indicated by indicator lights.
- There are two types of muting: parallel and sequential. This FB performs parallel muting with two muting sensors. An application example is provided below. (Refer to *SF_MutingPar_2Sensor Instruction Application Example with Two Reflective Light Barriers* on page 4-84.) The sensors must be located as described in *Annex F.7* in *CD 2005* of *IEC 62046,* as shown in the application example. The FB can be used for either forward or backward passage. However, the actual direction cannot be identified. To prevent manual operation, muting must also be enabled by process control with the *MutingEnable* signal to perform muting.
- The input parameters to the FB include two muting sensor signals (*S_MutingSwitch11* and *S_MutingSwitch12*), an OSSD signal from a photoelectric protection device (*S_AOPD_In*), and two time parameters (*DiscTimeEntry* and *MaxMutingTime*).
- Activate the *S_StartReset* input only when you can ensure that no hazardous state will occur as the result of starting the Safety CPU Unit.

● SF_MutingPar_2Sensor Instruction Application Example with Two Reflective Light Barriers

Order	Diagram	Description
1		If reflection light barriers are used as muting sensors, they are generally arranged diagonally. In general, this arrangement of reflection light barriers as muting sensors requires only two light barriers, and only *S_MutingSwitch11* (MS_11) and *S_MutingSwitch12* (MS_12) are allocated.

State Transition Diagram



Note 1. Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

2. A state transition to Error Muting sequence (priority 1), Error Timer (priority 2), Safety Demand AOPD (priority 3), or Error Muting lamp (priority 4) in the muting substates has higher priority than a state transition to muting substates with priority 5.

3. The muting conditions are described below.

Muting Conditions:

Muting Condition 1 (8000 to 8011) (when MS_11 is the first switch to start muting)
Timers for *DiscTimeEntry* and *MaxMutingTime* started: *MutingEnable* AND R_TRIG at MS_11 AND NOT MS_12

Muting Condition 2 (8000 to 8311) (when MS_12 is the first switch to start muting)
Timers for *DiscTimeEntry* and *MaxMutingTime* started: *MutingEnable* AND NOT MS_11 AND R_TRIG at MS_12

Muting Condition 3 (8011 to 8012) (when MS_12 is the second switch to start muting)
Timer for *DiscTimeEntry* stopped: *MutingEnable* AND MS_11 AND R_TRIG at MS_12

Muting Condition 4 (8311 to 8012) (when MS_11 is the second switch to start muting)
Timer for *DiscTimeEntry* stopped: *MutingEnable* AND R_TRIG at MS_11 AND MS_12

Muting Condition 5 (8000 to 8012) (when both switches simultaneously enable muting)
Timer for *MaxMutingTime* started: *MutingEnable* AND R_TRIG at MS_11 AND R_TRIG at MS_12

Muting Condition 6 (8012 to 8000) (when both switches simultaneously reset muting or when MS_11 and MS_12 are consecutively reset)
Timer for *MaxMutingTime* stopped: NOT MS_11 OR NOT MS_12

Illegal Muting Sequences:

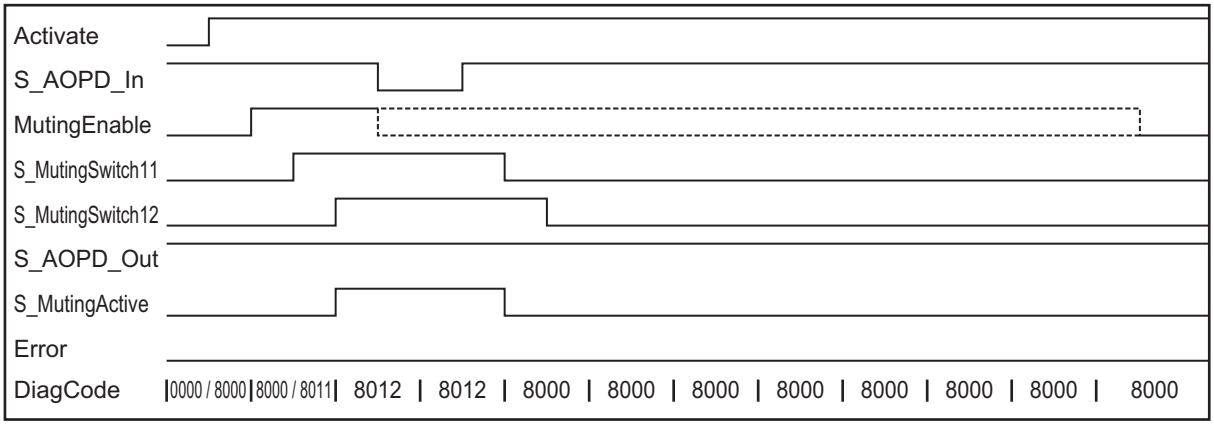
State 8000: (R_TRIG at MS_11 AND MS_12 AND NOT R_TRIG at MS_12) OR (R_TRIG at MS_12 AND MS_11 AND NOT R_TRIG at MS_11) OR ((MS_11 AND NOT R_TRIG at MS_11) AND (MS_12 AND NOT R_TRIG at MS_12)) OR (NOT *MutingEnable* AND R_TRIG at MS_11) OR (NOT *MutingEnable* AND R_TRIG at MS_12)

State 8011: NOT *MutingEnable* OR NOT MS_11

State 8311: NOT *MutingEnable* OR NOT MS_12

State 8012: All possible transitions allowed.

Timing Charts



S_StartReset = TRUE, Reset = FALSE, and S_MutingLamp = TRUE

Instruction Execution Errors

Error Detected

The FB detects the following errors.

- *DiscTimeEntry* is set to less than T#0s or more than T#4s.
- *MaxMutingTime* is set to less than T#0s or more than T#10min.
- The discrepancy time for the sensor pair (*S_MutingSwitch11* and *S_MutingSwitch12*) exceeded the set value.
- Muting (*S_MutingActive* = TRUE) exceeded *MaxMutingTime* (maximum muting time).
- The muting sensors (S_*MutingSwitch11* and *S_MutingSwitch12*) operated in an incorrect order.
- A muting sequence started without being enabled by *MutingEnable*.
- A muting sensor signal that is always TRUE was detected.
- A muting lamp failure was indicated (*S_MutingLamp* = FALSE).
- An undetected change to TRUE in the *Reset* input was detected in the Init state or the Wait for Reset state.

Operation for Errors

- When an error occurs, the *S_AOPD_Out* and *S_MutingActive* outputs are set to FALSE. The *DiagCode* output gives the relevant error code and the *Error* output is set to TRUE.
- Operation is not restarted until the error is reset and the operator acknowledges the safe state with *Reset*.

FB-specific Error Codes

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C001	49153	Reset Error 1	When the Init state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C002	49154	Reset Error 2	When the Wait for Reset state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE
C003	49155	Error Muting lamp	An error was detected in the muting lamp. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE
CYx4	--- *1	Error Muting sequence	An error was detected during muting sequence state 8000, 8011, or 8311. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE Y = Sequence status C0x4 = Error occurred in state 8000. C1x4 = Error occurred in state 8011. C2x4 = Error occurred in state 8311. CFx4 = *MutingEnable* was not detected. x = Muting sensor status when the error occurred (4 bits: LSB = MS_11; Next bit after LSB = MS_12). CY04: Both switches are FALSE. CY14: *S_MutingSwitch11* = TRUE CY24: *S_MutingSwitch12* =TRUE CY34: Both switches are TRUE.
C005	49157	Parameter Error	The value of *DiscTimeEntry* or *MaxMutingTime* is out of range. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE
C006	49158	Error Timer MaxMuting	Timing error: The muting operation time while *S_MutingActive* = TRUE exceeded *MaxMutingTime*. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE
C007	49159	Error Timer Entry	Timing error: Discrepancy time in *MutingSwitch11* and *MutingSwitch12* changing to TRUE exceeded *DiscTimeEntry.* *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE

*1. Find the DiagCode hexadecimal value with the information given in *Status description and output results* and then convert it to the DiagCode decimal value.

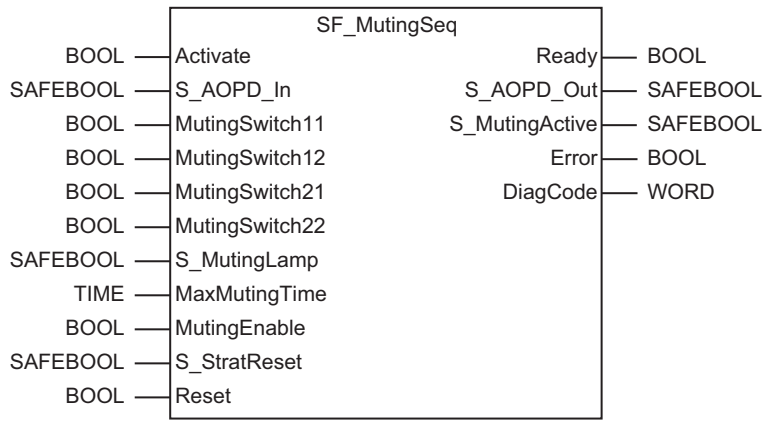
● FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = FALSE
8000	32768	AOPD Free	Muting is disabled but the control input from AOPD is active. If the muting timer is operating, it stops. *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = FALSE *Error* = FALSE
8001	32769	Init	The FB was started. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = FALSE
8002	32770	Safety Demand AOPD	Muting is disabled. The control input from AOPD is disabled. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = FALSE
8003	32771	Wait for Reset	A safety request or error was detected and cleared. The operator must respond with *Reset*. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = FALSE
8005	32773	Safe	The safety function is operating. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = FALSE
8011	32785	Muting Start 1	A muting sequence is being started after *Muting-Switch11* changed to TRUE. Monitoring is active for *DiscTimeEntry.* *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = FALSE *Error* = FALSE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8311	33553	Muting Start 2	A muting sequence is being started after *Muting-Switch12* changed to TRUE. Monitoring is active for *DiscTimeEntry.* *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = TRUE *Error* = FALSE
8012	32786	Muting Active	A muting sequence is in operation for one of the following. • A change to TRUE was detected in the second muting switch of *S_MutingSwitch11* and *S_MutingSwitch12*. • Both *S_MutingSwitch11* and *S_Muting-Switch12* were started in the same cycle. Monitoring for *DiscTimeEntry* was stopped. Monitoring for *MaxMutingTime* was started. *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = TRUE *Error* = FALSE

SF_MutingSeq

Muting is used to intentionally disable a safety function. This safety FB performs sequential muting with four muting sensors.

Instruction	Name	FB/FUN	Graphic expression
SF_MutingSeq	Sequential Muting	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_AOPD_In	SAFEBOOL	TRUE or FALSE	FALSE	This is the OSSD (safety output) signal from the AOPD (active optoelectronic protective device). FALSE: Something entered the protected area. TRUE: Nothing entered the protected area.
MutingSwitch11	BOOL*1	TRUE or FALSE	FALSE	A variable. It is the status of muting sensor 11. FALSE: Muting sensor 11 is not operating. TRUE: A workpiece activated muting sensor 11.
MutingSwitch12	BOOL*1	TRUE or FALSE	FALSE	A variable. It is the status of muting sensor 12. FALSE: Muting sensor 12 is not operating. TRUE: A workpiece activated muting sensor 12.
MutingSwitch21	BOOL*1	TRUE or FALSE	FALSE	A variable. It is the status of muting sensor 21. FALSE: Muting sensor 21 is not operating. TRUE: A workpiece activated muting sensor 21.
MutingSwitch22	BOOL*1	TRUE or FALSE	FALSE	A variable. It is the status of muting sensor 22. FALSE: Muting sensor 22 is not operating. TRUE: A workpiece activated muting sensor 22.
S_MutingLamp	SAFEBOOL	TRUE or FALSE	FALSE	A constant or a variable. It is the muting lamp status input (e.g., filament broken status). FALSE: Muting lamp failure. TRUE: Muting lamp is normal.
MaxMutingTime	TIME	T#0s to T#10min	T#0s	A constant. It sets the maximum time until completion of the muting sequence. The timer starts when the muting sensor first operates.

Variable	Data type	Valid range	Default	Description
MutingEnable	BOOL	TRUE or FALSE	FALSE	A constant or a variable. It is a command from the control system to enable starting muting as required in the machine cycle. You can change this signal to OFF after muting starts. FALSE: Disables muting. TRUE: Enables starting muting.
S_StartReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
Reset	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.

*1. You must connect a SAFEBOOL variable (not a BOOL variable) depending on safety requirements.

Output Variables

Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_AOPD_Out	SAFEBOOL	TRUE or FALSE	FALSE	This safety-related output gives the status of the protection devices that is being muted. FALSE: Something has entered the AOPD protected area and muting is disabled. TRUE: Nothing has entered the AOPD protected area and muting is enabled.
S_MutingActive	SAFEBOOL	TRUE or FALSE	FALSE	Gives the muting status. FALSE: Muting is disabled. TRUE: Muting is enabled.
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

Precautions for Correct Use

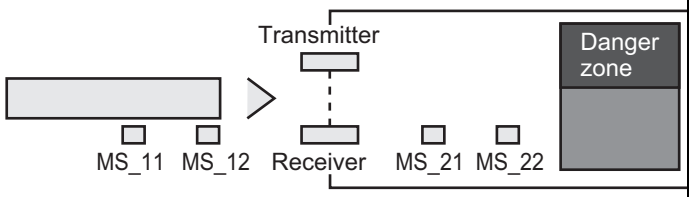
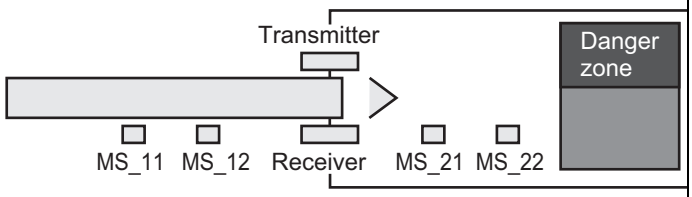
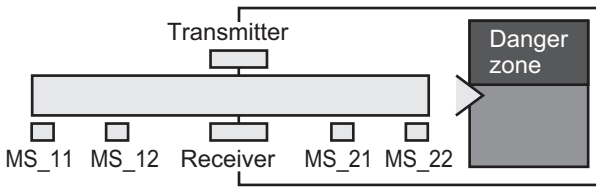
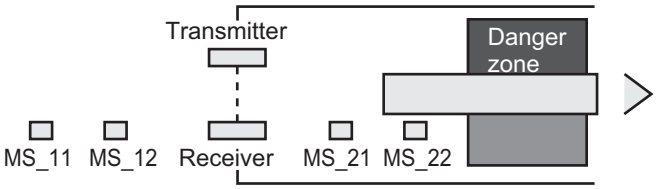
This FB does not detect short-circuits in muting sensor signals or errors in the function applications that supply those signals. It interprets them as illegal muting sequences. Unintentional muting must not be allowed under these conditions. Give attention to this during risk assessment.

Function

- Muting is used to intentionally disable a safety function. Muting is used, for example, to pass a workpiece through a hazardous area without stopping the machine. Muting is activated by muting sensors. Two or four muting sensors are used. To correctly incorporate the safety function into a manufacturing process, you must ensure that people will not enter the hazardous area while the light curtain is being muted. Proximity sensors, photoelectric barriers, limit switches, and other devices that do not have failsafe mechanisms are used for muting sensors. Muting operation must be indicated by indicator lights.
- There are two types of muting: parallel and sequential. This FB performs sequential muting with four muting sensors. Passing into a hazardous area in the forward direction is described below. (Refer to *SF_MutingSeq Instruction Application Example for Forward Entry with Four Sensors* on page 4-93.) The FB can be used for either forward or backward passage. To prevent manual operation, muting must also be enabled by process control with the *MutingEnable* signal to perform muting. If the *MutingEnable* signal is inactive, it must be set to TRUE.

- The input parameters to the FB include four muting sensor signals (*MutingSwitch11* to *MutingSwitch22*) and an OSSD signal from a photoelectric protection device (*S_AOPD_In*).
- Activate the *S_StartReset* input only when you can ensure that no hazardous state will occur as the result of starting the Safety CPU Unit.

● SF_MutingSeq Instruction Application Example for Forward Entry with Four Sensors

Order	Diagram	Description
1		Muting mode is enabled when the passage of a workpiece turns ON *MutingSwitch11* (MS_11) and then *MutingSwitch12* (MS_12).
2		Muting mode is enabled as long as *MutingSwitch11* (MS_11) and *MutingSwitch12* (MS_12) remain ON due to the workpiece. This allows the workpiece to pass through the light curtain without stopping the machine.
3		*MutingSwitch21* (MS_21) and *MutingSwitch22* (MS_22) must turn ON before *MutingSwitch11* (MS_11) and *MutingSwitch12* (MS_12) turn OFF. This ensures that muting mode remains ON.
4		Muting mode is ended if only *MutingSwitch22* (MS_22) is ON due to the workpiece.

Muting Conditions

Forward Direction

Muting Condition 1 (8000 to 8011) (when MS_11 is the first switch to enable muting)
Timer for *MaxMutingTime* started: *MutingEnable* AND (R_TRIG at MS_11 AND NOT MS_12 AND NOT MS_21 AND NOT MS_22)

Muting Condition 2 (8011 to 8012) (when MS_12 is the second switch to enable muting)
MutingEnable AND (MS_11 AND R_TRIG at MS_12 AND NOT MS_21 AND NOT MS_22)

Muting Condition 3 (8012 to 8000) (when MS_21 is the first switch to disable muting)
Timer for *MaxMutingTime* stopped: NOT MS_11 AND NOT MS_12 AND F_TRIG at MS_21 AND MS_22

Backward Direction

Muting Condition 11 (8000 to 8122) (when MS_22 is the first switch to enable muting)
Timer for *MaxMutingTime* started: *MutingEnable* AND (NOT MS_11 AND NOT MS_12 AND NOT MS_21 AND R_TRIG at MS_22)

Muting Condition 12 (8122 to 8112) (when MS_21 is the second switch to enable muting)
MutingEnable AND (NOT MS_11 AND NOT MS_12 AND R_TRIG at MS_21 AND MS_22)

Muting Condition 13 (8112 to 8000) (when MS_12 is the first switch to disable muting)
Timer for *MaxMutingTime* stopped: MS_11 AND F_TRIG at MS_12 AND NOT MS_21 AND NOT MS_22

Illegal Muting Sequences:

State 8000: (NOT *MutingEnable* AND R_TRIG at MS_11) OR (NOT *MutingEnable* AND R_TRIG at MS_22) OR (MS_12 OR MS_21) OR (MS_11 AND MS_22)

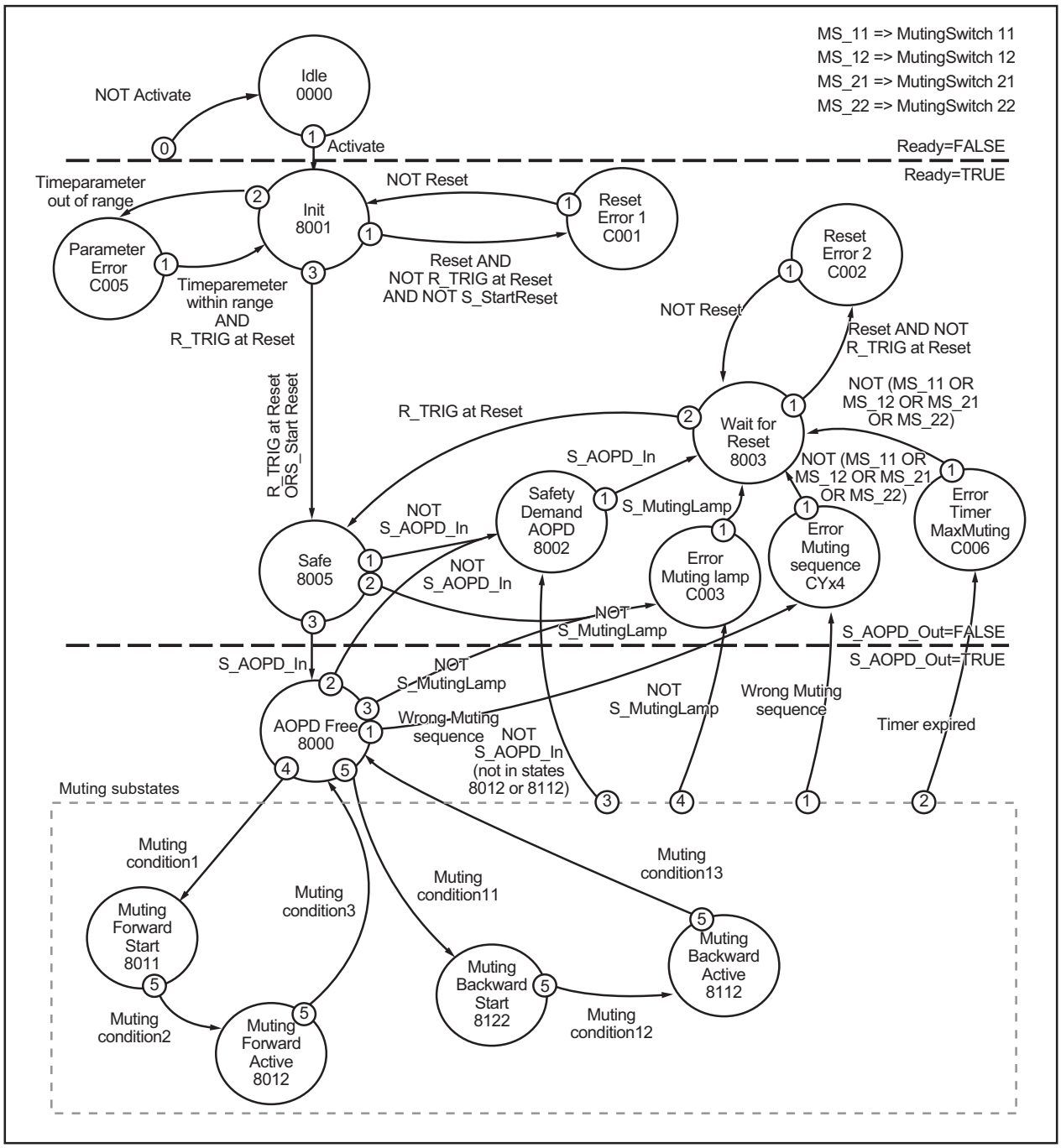
State 8011: NOT *MutingEnable* OR NOT MS_11 OR MS_21 OR MS_22

State 8012: R_TRIG at MS_11 OR R_TRIG at MS_12 OR F_TRIG at MS_22

State 8122: NOT *MutingEnable* OR MS_11 OR MS_12 OR NOT MS_22

State 8112: F_TRIG at MS_11 OR R_TRIG at MS_21 OR R_TRIG at MS_22

State Transition Diagram

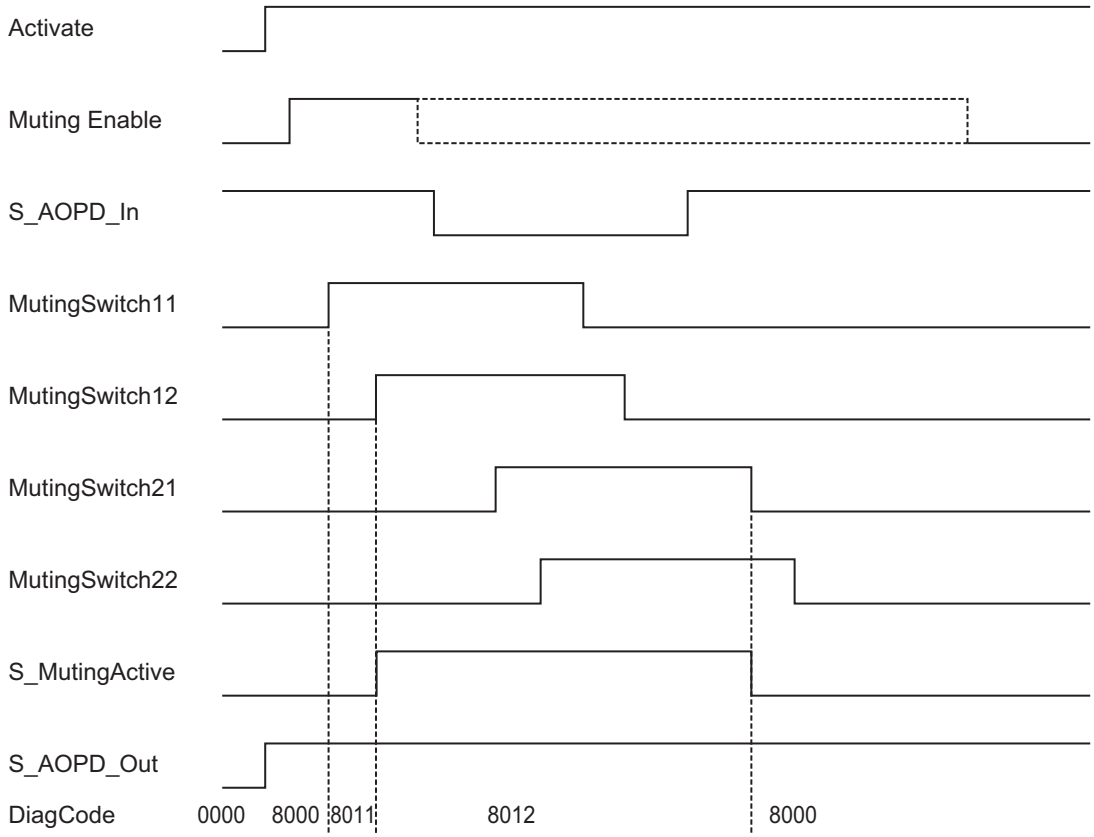


Note 1. Transitions to the Idle state from any other state are not shown for when Activate changes to FALSE. However, the transition to the Idle state has the highest priority (0).

2. A state transition to Error Muting sequence (priority 1), Error Timer (priority 2), Safety Demand AOPD (priority 3), or Error Muting lamp (priority 4) in the muting substates has higher priority than a state transition to muting substates with priority 5.

Timing Charts

SF_MutingSeq Instruction Timing Chart When *S_StartReset* = TRUE



Instruction Execution Errors

Error Detected

The FB detects the following errors.

- *MutingSwitch11*, *MutingSwitch12*, *MutingSwitch21*, and *MutingSwitch22* operated in an incorrect order.
- A muting sequence started without being enabled by *MutingEnable*.
- A muting lamp failure was indicated (*S_MutingLamp* = FALSE).
- An undetected change to TRUE in the *Reset* input was detected in state 8001 or 8003.
- *MaxMutingTime* is set to less than T#0s or more than T#10min.
- Muting (*S_MutingActive* = TRUE) exceeded *MaxMutingTime* (maximum muting time).

Operation for Errors

- When an error occurs, the *S_AOPD_Out and S_MutingActive* outputs are set to FALSE. The *DiagCode* output gives the relevant error code and the *Error* output is set to TRUE.
- Operation is not restarted until the error is reset and the operator acknowledges the safe state with *Reset*.

FB-specific Error Codes

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C001	49153	Reset Error 1	When the Init state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE
C002	49154	Reset Error 2	When the Wait for Reset state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE
C003	49155	Error Muting lamp	An error was detected in the muting lamp. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
CYx4	--- *1	Error Muting sequence	A muting sequence error was detected in state 8000, 8011, 8012, 8112, or 8122. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE Y = Sequence state (forward direction: 2 states, backward direction: 2 states) C0x4 = Error occurred in state 8000. C1x4 = Error occurred in forward direction in state 8011. C2x4 = Error occurred in forward direction in state 8012. C3x4 = Error occurred in backward direction in state 8122. C4x4 = Error occurred in backward direction in state 8112. CFx4 = *MutingEnable* was not detected. x = Sensor status when the error occurred (4 bits: LSB = MS_11; MS_12; MS_21; MSB = MS_22).
C005	49157	Parameter Error	The value of *MaxMutingTime* is outside of the valid range. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE
C006	49158	Error Timer MaxMuting	Timing error: The effective muting time while *S_MutingActive* = TRUE exceeded *MaxMuting-Time*. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = TRUE

*1. Find the DiagCode hexadecimal value with the information given in *Status description and output results* and then convert it to the DiagCode decimal value.

● FB-specific State Codes (No Error)

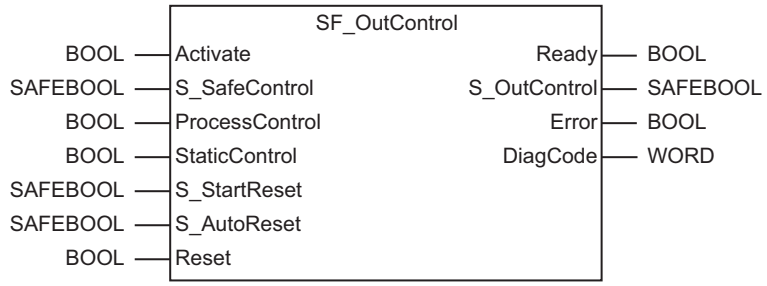
DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = FALSE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8000	32768	AOPD Free	Muting is disabled. The control input from AOPD is active. *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = FALSE *Error* = FALSE
8001	32769	Init	The FB was started. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = FALSE
8002	32770	Safety Demand AOPD	Muting is disabled. The control input from AOPD is disabled. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = FALSE
8003	32771	Wait for Reset	A safety request or error was detected and cleared. The operator must respond with *Reset*. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = FALSE *Error* = FALSE
8005	32773	Safe	The safety function is operating. *Ready* = TRUE *S_AOPD_Out* = FALSE *S_MutingActive* = TRUE *Error* = FALSE
8011	32785	Muting Forward Start	A forward muting sequence is being started and there is no safety request. *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = FALSE *Error* = FALSE
8012	32786	Muting Forward Active	A forward muting sequence is in operation. *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = TRUE *Error* = FALSE
8112	33042	Muting Backward Active	A backward muting sequence is in operation. *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = TRUE *Error* = FALSE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8122	33058	Muting Backward Start	A backward muting sequence is being started and there is no safety request. *Ready* = TRUE *S_AOPD_Out* = TRUE *S_MutingActive* = FALSE *Error* = FALSE

SF_OutControl

This safety FB controls a safety output with a control signal and safety signal from a function application.

Instruction	Name	FB/FUN	Graphic expression
SF_OutControl	Out Control	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_SafeControl	SAFEBOOL	TRUE or FALSE	FALSE	A variable. It is the control signal from the previous safety FB. Use the signal from a typical FB from the library (SF_EmergencyStop, SF_GuardMonitoring, SF_TwoHandControlTypeII, etc.) FALSE: The signal from the previous safety FB is inactive. TRUE: The signal from the previous safety FB is active.
ProcessControl	BOOL	TRUE or FALSE	FALSE	A constant or a variable. It is a control signal from the function application. FALSE: Request to set *S_OutControl* to FALSE. TRUE: Request to set *S_OutControl* to TRUE.
StaticControl	BOOL	TRUE or FALSE	FALSE	A constant. It is the process control option status. FALSE: A change to TRUE in *ProcessControl* is required after the FB is started or after the safety function is triggered. TRUE: A change to TRUE in *ProcessControl* is not required after the FB is started or after the safety function is triggered.
S_StartReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_AutoReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
Reset	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.

Output Variables

Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_OutControl	SAFEBOOL	TRUE or FALSE	FALSE	Controls the connected actuator. FALSE: Disables the connected actuator. TRUE: Enables the connected actuator.
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

Function

Introduction

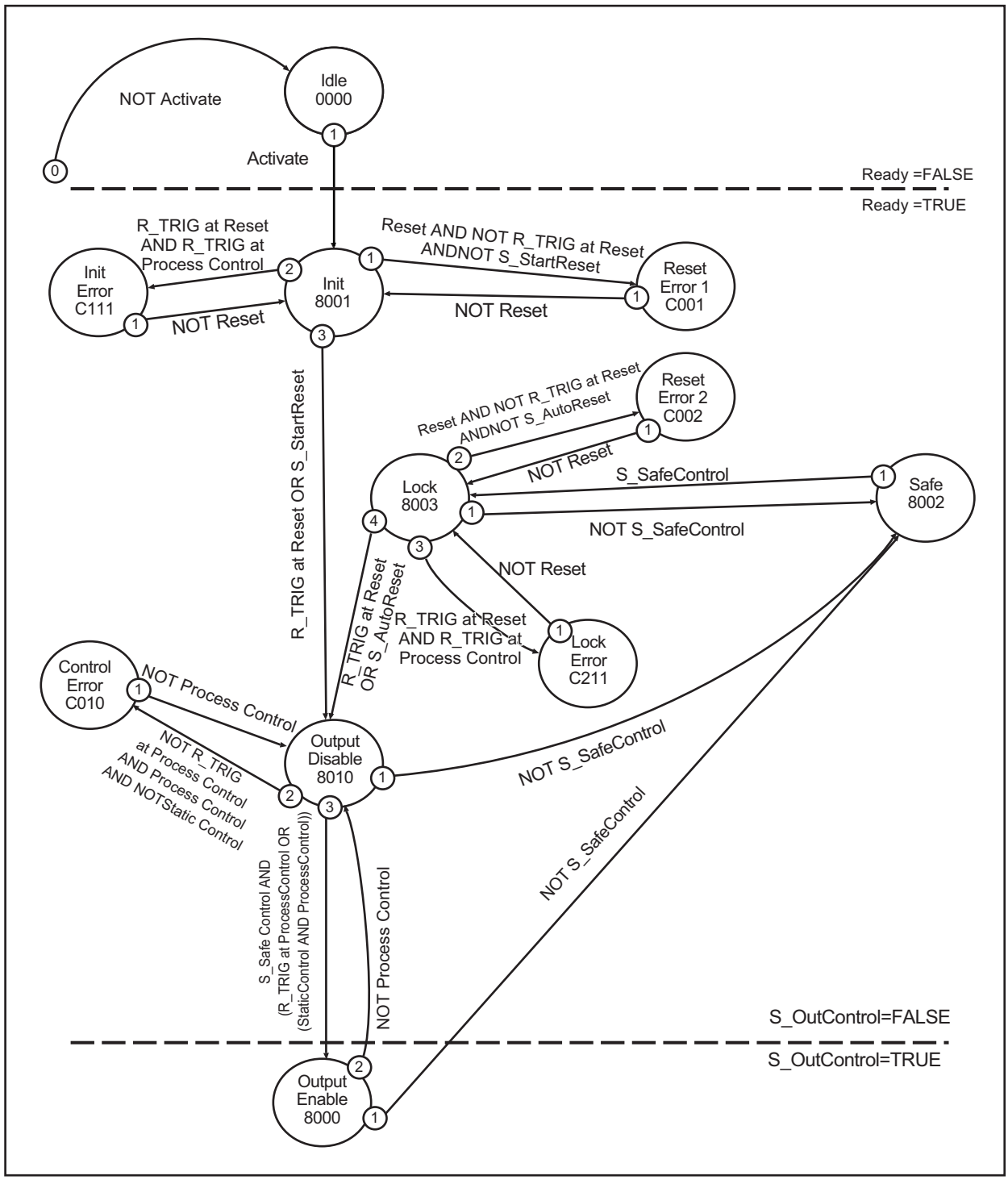
This FB is an output drive device for a safety output.
The safety output is controlled through *S_OutControl* using a signal from the function application (ProcessControl/BOOL to control the process) and a signal from the safety application (S_SafeControl/SAFEBOOL to control the safety function).

Process Control Option Status (StaticControl)

- If *StaticControl* is FALSE, the function must be started again by changing *ProcessControl* to TRUE after the FB starts or after the safety signal (*S_SafeControl*) feedback. An always-TRUE signal for *ProcessControl* will not set *S_OutControl* to TRUE.
- If *StaticControl* is TRUE, it is not necessary to start the function again by changing *ProcessControl* to TRUE after the FB starts or after the safety signal (*S_SafeControl*) feedback. As long as the other conditions are met, an always-TRUE signal for *ProcessControl* will set *S_OutControl* to TRUE.

Activate the *StaticControl, S_StartReset,* and *S_AutoReset* inputs only when you can ensure that no hazardous state will occur as the result of starting the Safety CPU Unit.

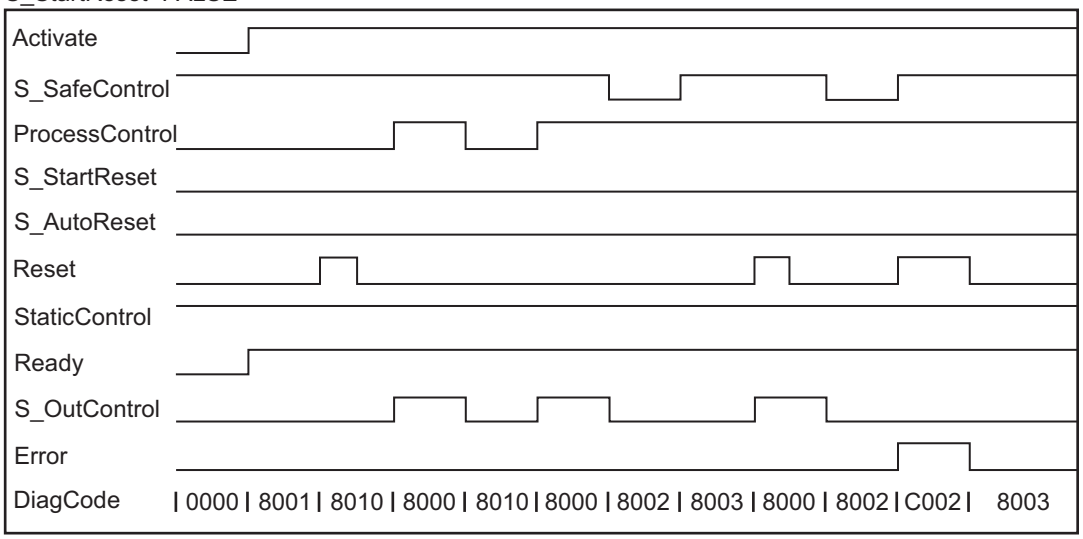
State Transition Diagram



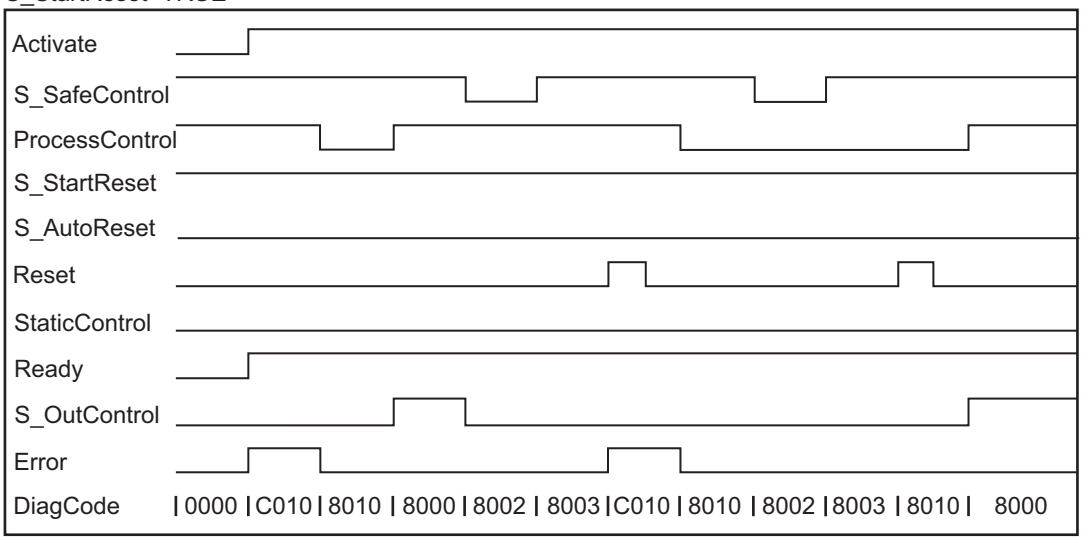
Note Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

Timing Charts

S_StartReset=FALSE



S_StartReset=TRUE



Instruction Execution Errors

Error Detected

The following conditions force a transition to an error state.

- An invalid process always-TRUE *Reset* signal
- An invalid always-TRUE *ProcessControl* signal
- Programming error that results in incorrect interconnections between *ProcessControl* and *Reset*

Operation for Errors

- If an error occurs, the *S_OutControl* output changes to FALSE and the safe state is maintained.
- To leave the Reset, Init, or Lock error states, you must set the *Reset* input to FALSE. To leave the Control Error state, you must set the *ProcessControl* input to FALSE.
- After *S_SafeControl* changes to TRUE, you can change the *Reset* input to TRUE to reset the optional startup inhibit. Or, after the FB is started, you can change the *Reset* input to TRUE to reset the optional startup inhibit.

FB-specific Error Codes

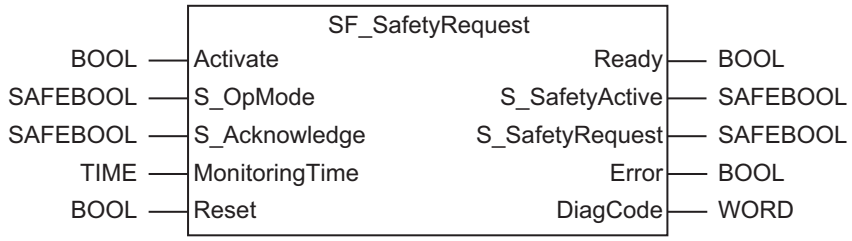
DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C001	49153	Reset Error 1	When the Init state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_OutControl* = FALSE *Error* = TRUE
C002	49154	Reset Error 2	When the Lock state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_OutControl* = FALSE *Error* = TRUE
C010	49168	Control Error	When output was disabled, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_OutControl* = FALSE *Error* = TRUE
C111	49425	Init Error	*Reset* and *ProcessControl* simultaneously changed to TRUE in state 8001. *Ready* = TRUE *S_OutControl* = FALSE *Error* = TRUE
C211	49681	Lock Error	*Reset* and *ProcessControl* simultaneously changed to TRUE in state 8003. *Ready* = TRUE *S_OutControl* = FALSE *Error* = TRUE

● FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *S_OutControl* = FALSE *Error* = FALSE
8001	32769	Init	*Activate* is set to TRUE and the FB is activated. *S_StartReset* is ON. Resetting is required. *Ready* = TRUE *S_OutControl* = FALSE *Error* = FALSE
8002	32770	Safe	*S_SafeControl* changed to OFF, so *S_OutControl* changed to OFF. *Ready* = TRUE *S_OutControl* = FALSE *Error* = FALSE
8003	32771	Lock	*S_SafeControl* changed to ON and *S_AutoReset* changed to OFF, so the FB is waiting for a *Reset* input. *Ready* = TRUE *S_OutControl* = FALSE *Error* = FALSE
8010	32784	Output Disable	*ProcessControl* is OFF. *Ready* = TRUE *S_OutControl* = FALSE *Error* = FALSE
8000	32768	Output Enable	*ProcessControl* is ON. *Ready* = TRUE *S_OutControl* = TRUE *Error* = FALSE

SF_SafetyRequest

This safety FB makes requests for the safe state and monitors the safety state for an actuator (e.g., a drive or valve) that has a safety function.

Instruction	Name	FB/FUN	Graphic expression
SF_SafetyRequest	Safety Request	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_OpMode	SAFEBOOL	TRUE or FALSE	FALSE	A variable. It is a request for the operation mode of the connected actuator. FALSE: Requests safe mode. TRUE: Requests an operation mode (i.e., non-safe state).
S_Acknowledge	SAFEBOOL	TRUE or FALSE	FALSE	A variable. It is the response from the connected actuator (to confirm whether the actuator is in safe mode). FALSE: Operation mode (i.e., non-safe state). TRUE: Safe mode.
MonitoringTime	TIME	Depends on data type.	T#0s	A constant. It inputs the monitoring time from the safe mode request (i.e., from when *S_OpMode* changed to FALSE) until the actuator response (i.e., when *S_Acknowledge* changes to TRUE).
Reset	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.

Output Variables

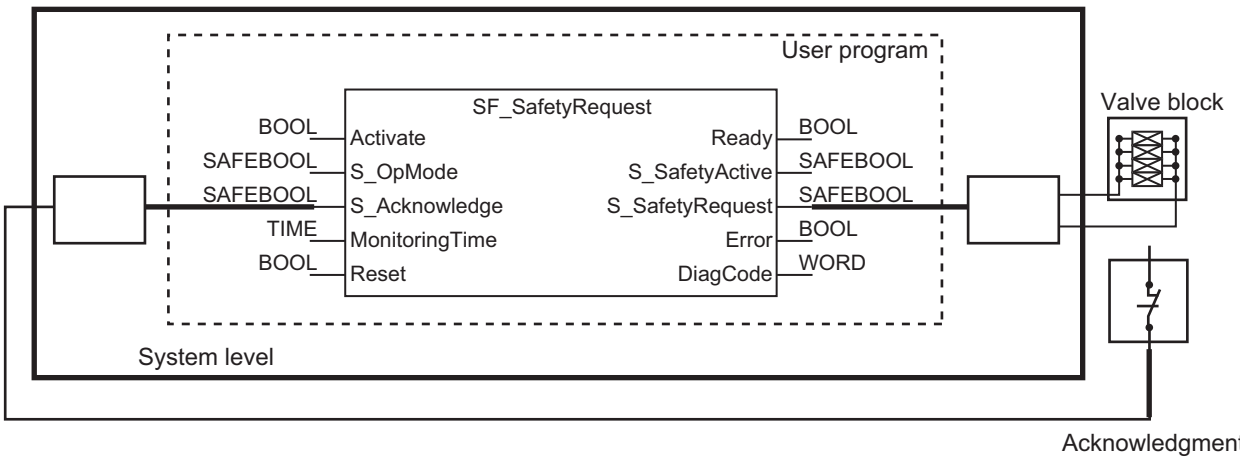
Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_SafetyActive	SAFEBOOL	TRUE or FALSE	FALSE	The actuator operation mode. FALSE: Non-safe state. TRUE: Safe mode.
S_SafetyRequest	SAFEBOOL	TRUE or FALSE	FALSE	The operation mode request to the actuator. FALSE: Requests safe mode. TRUE: Requests an operation mode (i.e., non-safe state).
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.

Variable	Data type	Valid range	Default	Description
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

Function

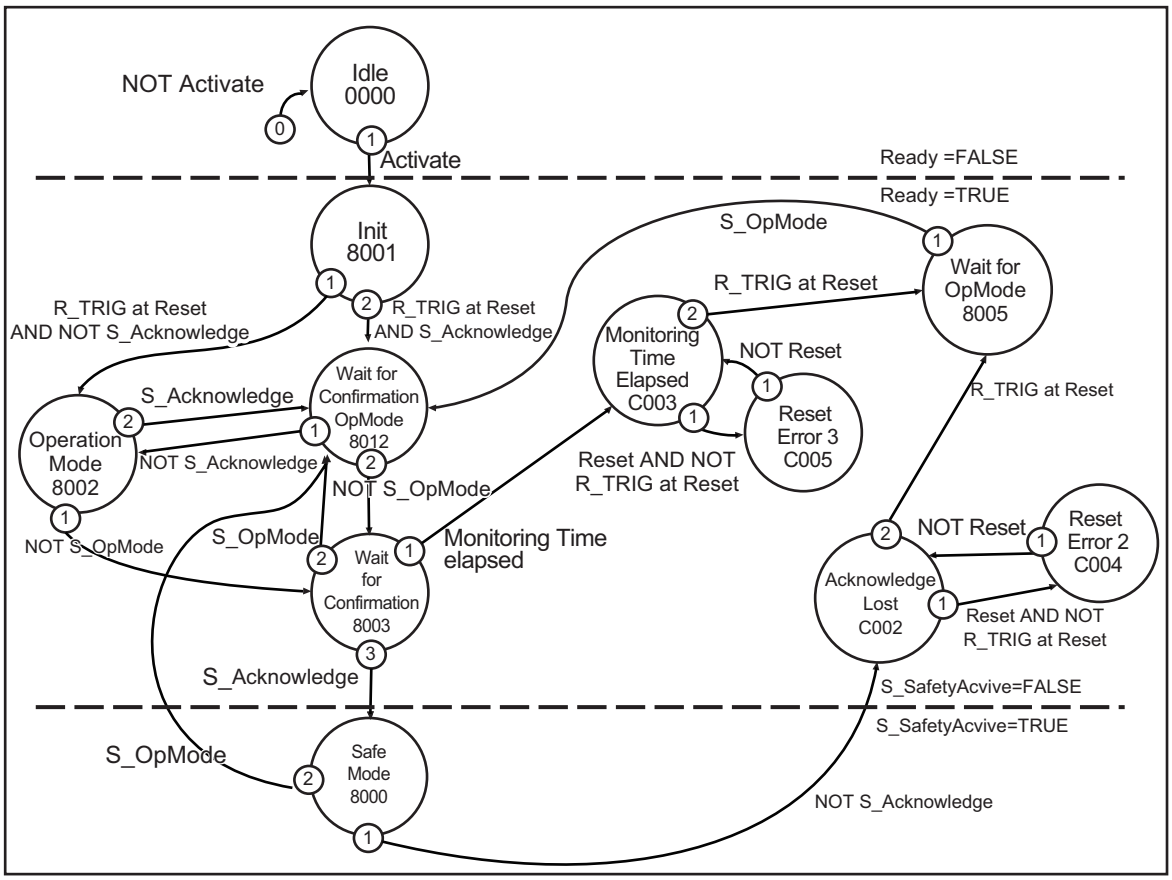
- This FB sends a request to change to safe mode to an actuator that has a safe mode and monitors the status.
- *S_SafetyRequest* is used to send a request to change to safe mode to the actuator according to the mode given by *S_OpMode.*
- The actuator returns the results of attempting to change to safe mode to *S_Acknowledge*.
- The FB outputs *S_SafetyActive* if the actuator changes to safe mode within the monitoring time (*MonitoringTime*) from when the safe mode request was made.

This FB is used as an interface between the safety-related system and an actuator. This means that the actuator's safety-related functions can be used in the application program. However, there are only two binary signals (the request signal and acknowledge signal) that are used to control the safe state of the actuator.



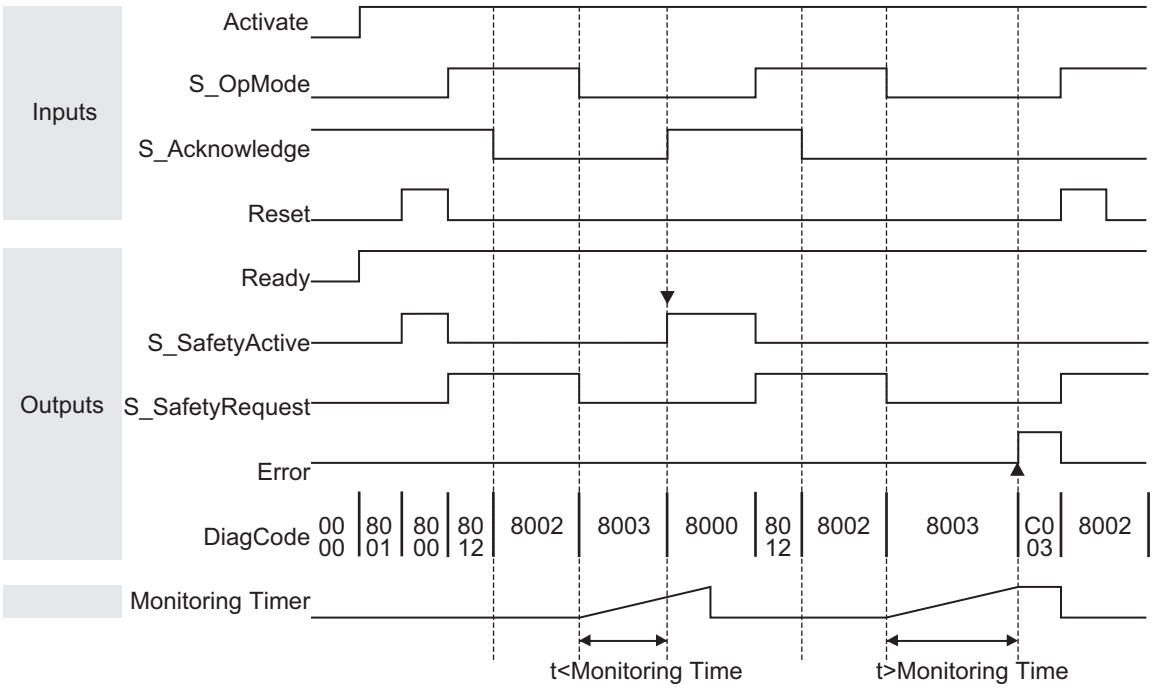
- This safety function is provided by the actuator. Therefore, the FB only starts a request, monitors the request, and sets an output after confirming the safe state of the actuator. This output is given by the *S_SafetyActive* output.
- This FB does not define actuator-specific parameters. The parameters must be defined in the actuator. The FB changes the actuator from operation mode to the safe state.

State Transition Diagram



Note Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

Timing Charts



Instruction Execution Errors

Error Detected

The FB detects the following errors.

- When the actuator does not enter the safe state within the monitoring time
- When the acknowledge signal is not sent before the request becomes invalid
- For an always-TRUE *Reset* signal
- When an undetected change to TRUE in the *Reset* input is detected when the acknowledge signal is lost or the monitoring time is exceeded

FB External Errors:

There are no external errors because error bits and error information are not provided by a normal actuator.

Operation for Errors

- When an error occurs, the *S_SafetyActive* output is set to FALSE.
- Acknowledgment by changing the *Reset* input to TRUE is required for an error. After this reset, the *S_OpMode* request must be set to TRUE to enable the FB to continue functioning.

FB-specific Error Codes

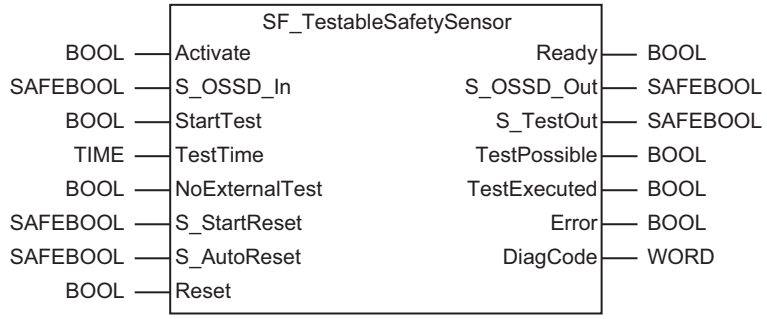
DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C002	49154	Acknowledge Lost	The acknowledge signal was lost in the Safe state. *Ready* = TRUE *S_SafetyActive* = FALSE *S_SafetyRequest* = FALSE *Error* = TRUE
C003	49155	MonitoringTime Elapsed	The *S_OpMode* input request was not completed within *MonitoringTime*. *Ready* = TRUE *S_SafetyActive* = FALSE *S_SafetyRequest* = FALSE *Error* = TRUE
C004	49156	Reset Error 2	When the Acknowledge Lost state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_SafetyActive* = FALSE *S_SafetyRequest* = FALSE *Error* = TRUE
C005	49157	Reset Error 3	When the Monitoring Time Elapsed state was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_SafetyActive* = FALSE *S_SafetyRequest* = FALSE *Error* = TRUE

● FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *S_SafetyActive* = FALSE *S_SafetyRequest* = FALSE *Error* = FALSE
8000	32768	Safe Mode	The actuator is in safe mode. *Ready* = TRUE *S_SafetyActive* = TRUE *S_SafetyRequest* = FALSE *Error* = FALSE
8001	32769	Init	*Activate* was set to TRUE and then *Reset* was set to TRUE. *Ready* = TRUE *S_SafetyActive* = FALSE *S_SafetyRequest* = FALSE *Error* = TRUE
8002	32770	Operation Mode	An operation mode exists for which safe mode cannot be confirmed. *Ready* = TRUE *S_SafetyActive* = FALSE *S_SafetyRequest* = TRUE *Error* = FALSE
8012	32786	Wait for Confirmation OpMode	An operation mode exists for which safe mode was confirmed. *Ready* = TRUE *S_SafetyActive* = FALSE *S_SafetyRequest* = TRUE *Error* = FALSE
8003	32771	Wait for Confirmation	The FB is waiting for acknowledgment from the drive device (i.e., the system interface). *Ready* = TRUE *S_SafetyActive* = FALSE *S_SafetyRequest* = FALSE *Error* = TRUE
8005	32773	Wait for OpMode	An error was reset, but *S_OpMode* must be set to TRUE before the FB is initialized. *Ready* = TRUE *S_SafetyActive* = FALSE *S_SafetyRequest* = FALSE *Error* = TRUE

SF_TestableSafetySensor

This safety FB tests functionality with the external test function of electro-sensitive protective equipment (ESPE). For example, it detects the loss of sensing unit detection ability, response times that exceed specified values, and always-ON signals from a single-channel sensor system. It can be used with a safety sensor that supports external testing (ESPE: electro-sensitive protective equipment, such as a light beam).

Instruction	Name	FB/FUN	Graphic expression
SF_TestableSafe-tySensor	Testable Safety Sensors	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_OSSD_In	SAFEBOOL	TRUE or FALSE	FALSE	A variable. It is the status of the sensor output. Example: Light curtain FALSE: The safety sensor is in test status or there is a request for a safety-related response. TRUE: The sensor is in normal operating condition.
StartTest	BOOL	TRUE or FALSE	FALSE	A variable. It is the sensor test start input. *S_TestOut* is set and the internal time monitor in the FB is started. FALSE: There is no test request. TRUE: There is a test request.
TestTime	TIME	0 to 150 ms	T#10ms	A constant. It inputs the test monitoring time for the safety sensor.
NoExternalTest	BOOL	TRUE or FALSE	FALSE	A constant. It indicates if a manual external test is supported for the sensor. FALSE: A manual external test is supported. If an error occurs in the automatic sensor test, an external manual sensor test is required. An automatic test will be possible again only after a manual sensor test sequence is completed. TRUE: A manual external test is not supported. If an error occurs in the automatic sensor test, an automatic test is possible again without a manual sensor test.
S_StartReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_AutoReset	SAFEBOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
Reset	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.

Output Variables

Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_OSSD_Out	SAFEBOOL	TRUE or FALSE	FALSE	The safety-related output that gives the ESPE status. FALSE: There is a safety-related operation request for the sensor or a test error occurred. TRUE: There is no safety-related operation request for the sensor and no test error occurred. Note OSSD is an output signal switching device.
S_TestOut	SAFEBOOL	TRUE or FALSE	FALSE	Forms a pair with the sensor test input. Although SAFEBOOL is specified, this signal is commonly connected to a BOOL output. FALSE: There is a test request. TRUE: There is no test request.
TestPossible	BOOL	TRUE or FALSE	FALSE	The feedback signal to the process. FALSE: An automatic sensor test cannot be performed. TRUE: An automatic sensor test can be performed.
TestExecuted	BOOL	TRUE or FALSE	FALSE	When the signal changes to TRUE, the automatic sensor test was executed normally. FALSE: • An automatic sensor test was not performed. • An automatic sensor test is active. • The automatic sensor test failed. TRUE: The sensor test was executed normally.
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

Function

- You can use this FB to execute a test for a type-2 ESPE sensor that has an external test function.
- The test simulates the operation of the sensing device and detects potentially hazardous problems (e.g., loss of sensing unit detection ability and response times that exceed specified values).
- During the test, the FB holds the safety output (*S_OSSD_Out*), so the test can be performed without stopping the safety output to the actuator.
- The FB simulates an entry into the hazardous area of a safety sensor that has an external test function (e.g., type-2 ESPE), and monitors the operation and the maximum response time.
- It is assumed that an external safety-related control system (e.g., machine) starts a periodic test. An ESPE must be connected to an applicable input device (e.g., safety input terminal).
- You must select the ESPE according to the required category in product specifications IEC 61496-1, IEC 61496-2, IEC 61496-3, and IEC 13849.
- You must monitor testing with a separate mechanism to ensure that the test is started at a suitable interval.
- Activate the *S_StartReset* and *S_AutoReset* inputs only when you can ensure that no hazardous state will occur as the result of starting the Safety CPU Unit.

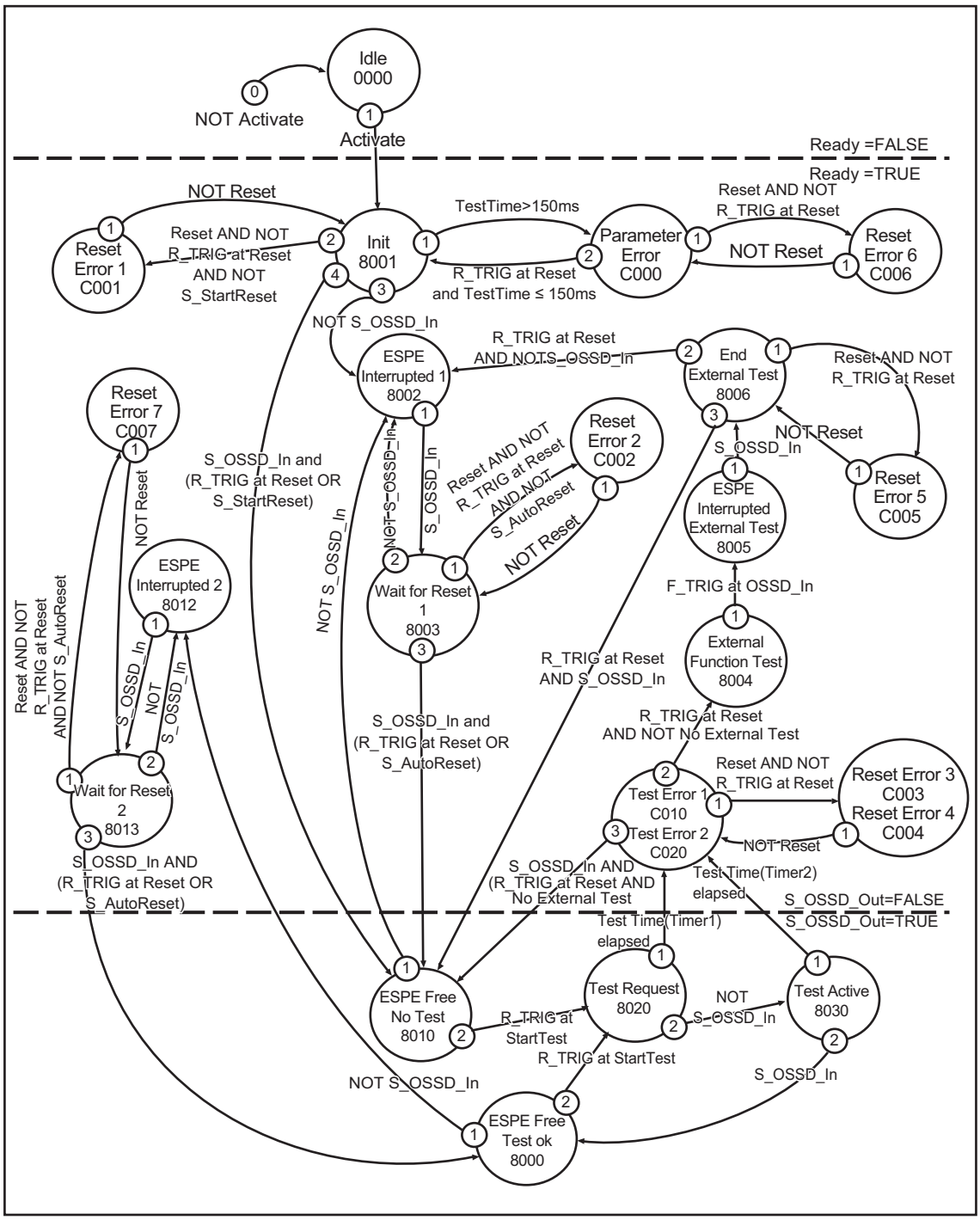
● Test Mode

1. When the *StartTest* input changes to TRUE, the *S_TestOut* output is set to FALSE and the internal monitor is started.
2. The *S_TestOut* signal stops light emission. (Monitoring for the first *TestTime* starts.)
3. *S_OSSD_In* changes to FALSE. (Monitoring for the second *TestTime* starts.)
4. *S_TestOut* changes to TRUE.
5. Light emission from the emitter starts.
6. The *S_OSSD_In* sensor input changes to TRUE.
7. The monitoring time is stopped.
8. *S_OSSD_Out* is set to TRUE during the test.

● Startup Control Options

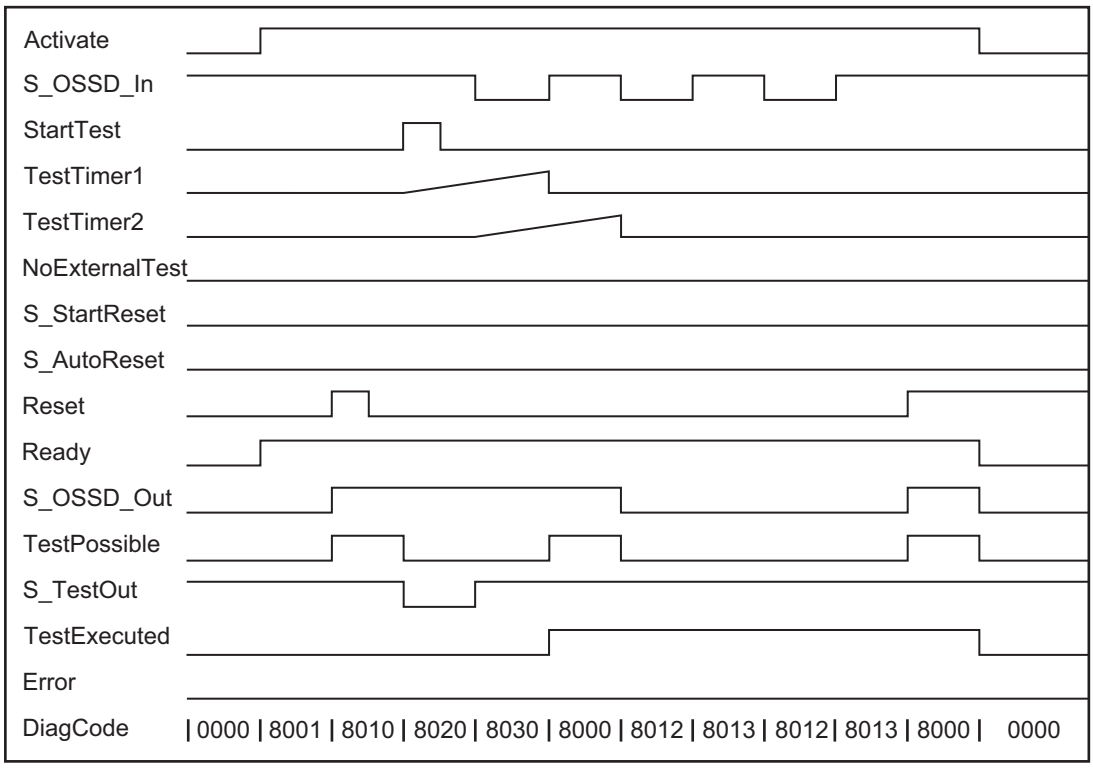
- Control starts after the FB is activated.
- Control starts after a protective device interrupt.

State Transition Diagram



Note Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

Timing Charts



Instruction Execution Errors

Error Detected

The following conditions force a transition to an error state.

- The test time was exceeded without sensor feedback extension.
- A test without sensor signal feedback
- Invalid always-TRUE *Reset* signal during processing
- Plausibility check of the monitoring time setting

Operation for Errors

- When an error occurs, the *S_OSSD_Out* output is set to FALSE and the safe state is maintained.
- When the error is removed, the sensor turns ON (*S_OSSD_In* = TRUE). When the *Reset* input changes to TRUE, FB error status is reset and the *S_OSSD_Out* output is set to TRUE.
- If *S_AutoReset* is FALSE, a request to make it TRUE is made when the FB is reset.
- After *S_OSSD_In* changes to TRUE, you can reset the FB by making the *Reset* input TRUE.
- After the FB starts, you can reset the FB by making the *Reset* input TRUE.

FB-specific Error Codes

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C000	49152	Parameter Error	The *TestTime* parameter is set to an invalid value. The setting range is from 0 to 150 ms. *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = TRUE
C001	49153	Reset Error 1	When the FB is activated, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = FALSE
C002	49154	Reset Error 2	When state 8003 (Wait for Reset 1) was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = TRUE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C003	49155	Reset Error 3	When state C010 (Test Error 1) was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = TRUE
C004	49156	Reset Error 4	When state C020 (Test Error 2) was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = TRUE
C005	49157	Reset Error 5	When state 8006 (End External Test) was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = TRUE
C006	49158	Reset Error 6	When state C000 (Parameter Error) was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = TRUE
C007	49159	Reset Error 7	When state 8013 (Wait for Reset 2) was entered, an undetected change to TRUE in the *Reset* input was detected. *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = TRUE *Error* = TRUE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C010	49168	Test Error 1	The test time was exceeded in state 8020 (Test Request). *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = TRUE
C020	49184	Test Error 2	The test time was exceeded in state 8030 (Test Active). *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = TRUE

● FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = FALSE
8001	32769	Init	The FB detected an activate signal and the FB is active. *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = FALSE
8002	32770	ESPE Interrupted 1	The FB detected a safety request. The switch has not been automatically tested. *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = FALSE

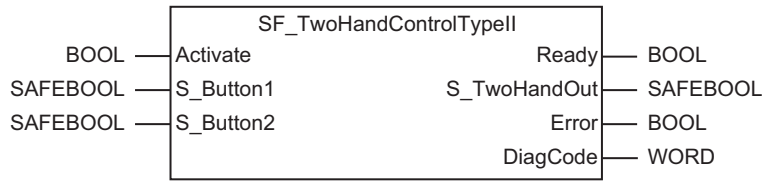
DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8003	32771	Wait for Reset 1	The FB is waiting for *Reset* to change to TRUE after state 8002. *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = FALSE
8004	32772	External Function Test	The automatic sensor test failed. An external manual sensor test is required. The FB started support for the required external manual sensor test (*NoExternalTest* = FALSE). The sensor signal must be made FALSE. *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = FALSE
8005	32773	ESPE Interrupted External Test	The automatic sensor test failed. An external manual sensor test is required. The FB started support for the required external manual sensor test (*NoExternalTest* = FALSE). The sensor signal must be TRUE. *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = FALSE
8006	32774	End External Test	The automatic sensor test failed. An external manual sensor test is required. The FB started support for the required external manual sensor test (*NoExternalTest* = FALSE). The external manual test was completed. The FB detected the completion of the sensor switching cycle (external control). *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = FALSE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8010	32784	ESPE Free No Test	*S_OSSD_In* is set to TRUE (the AOPD is receiving light). A sensor test has not been performed. *Ready* = TRUE *S_OSSD_Out* = TRUE *S_TestOut* = TRUE *TestPossible* = TRUE *TestExecuted* = FALSE *Error* = FALSE
8020	32800	Test Request	An automatic sensor test is in progress. The FB is waiting for the signal from the sensor to change to FALSE. The time from when a test was requested from the sensor until the sensor signal changes to FALSE is being monitored. *Ready* = TRUE *S_OSSD_Out* = TRUE *S_TestOut* = FALSE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = FALSE
8030	32816	Test Active	An automatic sensor test is in progress. The FB is waiting for the signal from the sensor to change to FALSE. The time from when a test was requested from the sensor until the sensor signal changes to TRUE is being monitored. *Ready* = TRUE *S_OSSD_Out* = TRUE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = FALSE *Error* = FALSE
8000	32768	ESPE Free Test ok	The FB did not detect a safety request. The sensor was tested automatically. *Ready* = TRUE *S_OSSD_Out* = TRUE *S_TestOut* = TRUE *TestPossible* = TRUE *TestExecuted* = TRUE *Error* = FALSE
8012	32786	ESPE Interrupted 2	The FB detected a safety request. The sensor was tested automatically. *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = TRUE *Error* = FALSE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8013	32787	Wait for Reset 2	The FB is waiting for *Reset* to change to TRUE after the EPSE Interrupted 2 state. *Ready* = TRUE *S_OSSD_Out* = FALSE *S_TestOut* = TRUE *TestPossible* = FALSE *TestExecuted* = TRUE *Error* = FALSE

SF_TwoHandControlTypeII

This safety FB provides a type II, two-hand control function as defined in ISO 13851 (EN 574).

Instruction	Name	FB/FUN	Graphic expression
SF_TwoHandControlTypeII	Two-Hand Control Type II	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_Button1	SAFEBOOL	TRUE or FALSE	FALSE	A variable. It is the button 1 input (category 3 or 4: for two antivalent contacts). FALSE: Button 1 is OFF. TRUE: Button 1 is ON.
S_Button2	SAFEBOOL	TRUE or FALSE	FALSE	A variable. It is the button 2 input (category 3 or 4: for two antivalent contacts). FALSE: Button 2 is OFF. TRUE: Button 2 is ON.

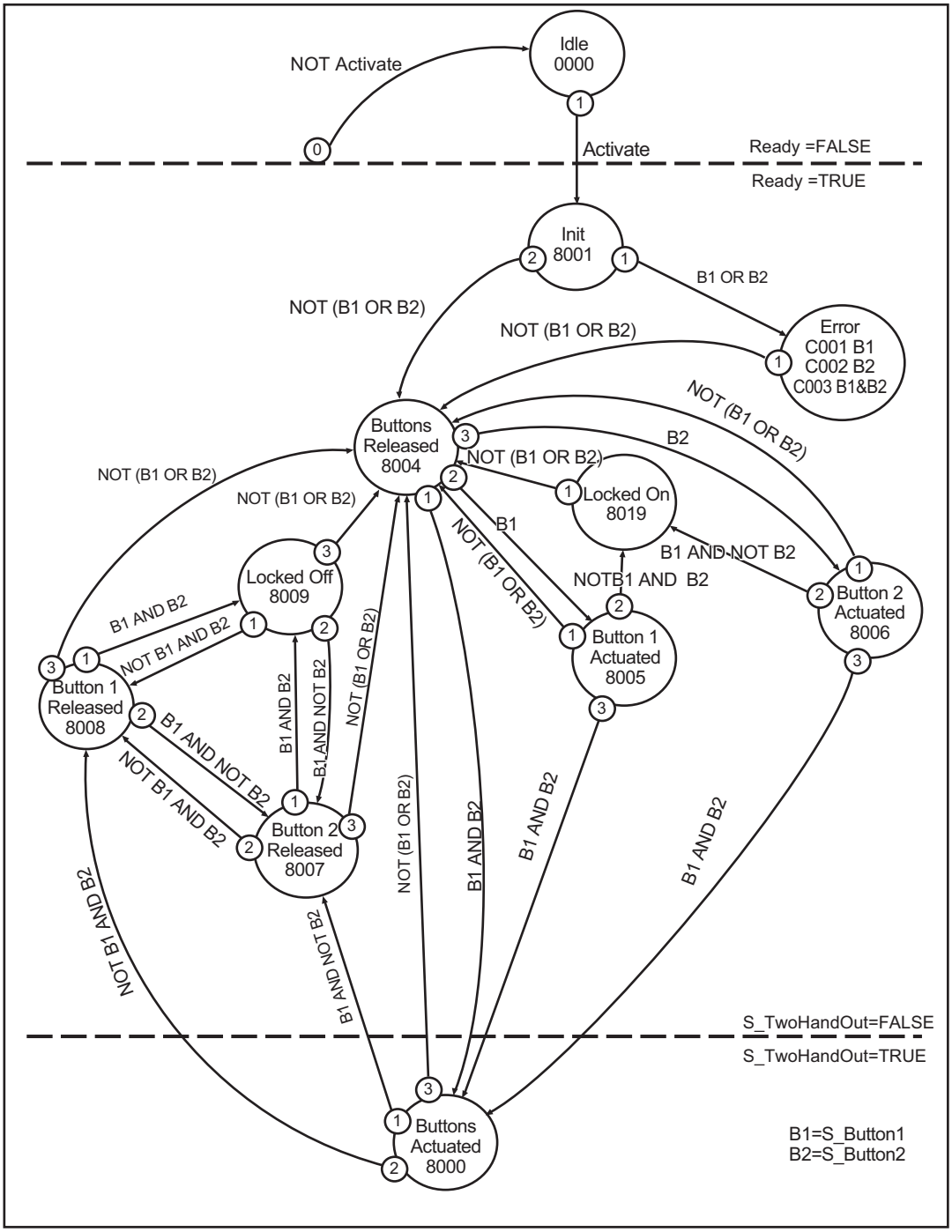
Output Variables

Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_TwoHandOut	SAFEBOOL	TRUE or FALSE	FALSE	The safety output signal. FALSE: A button is not being operated or correct two-hand operation was not performed. TRUE: The *S_Button1* and *S_Button2* inputs are TRUE and there is no error. Correct two-hand operation is being performed.
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

Function

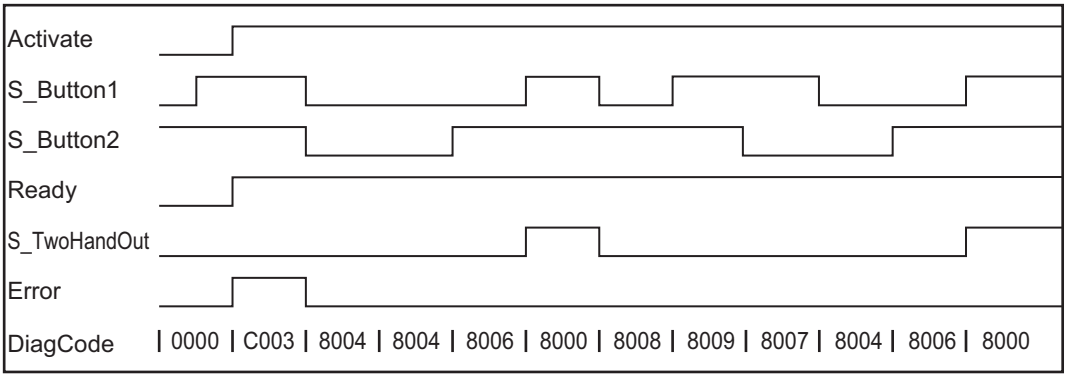
- This FB provides a type II, two-hand control function as defined in ISO 13851 (EN 574). If *S_Button1* and *S_Button2* are set to TRUE in the correct order, the *S_TwoHandOut* output is also set to TRUE.
- This FB also controls releasing of both buttons before it sets the *S_TwoHandOut* output to TRUE again.

State Transition Diagram



Note Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

Timing Charts



Instruction Execution Errors

Error Detected

After the FB is activated, it detects buttons that are already set to TRUE as illegal input settings that result in errors.

Operation for Errors

- When an error occurs, the *S_TwoHandOut* output is set to FALSE and the safe state is maintained.
- When both buttons are released (i.e., set to FALSE), the error status is reset.

FB-specific Error Codes

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C001	49153	Error B1	*S_Button1* was TRUE when the FB was activated. *Ready* = TRUE *Error* = TRUE *S_TwoHandOut* = FALSE
C002	49154	Error B2	*S_Button2* was TRUE when the FB was activated. *Ready* = TRUE *Error* = TRUE *S_TwoHandOut* = FALSE
C003	49155	Error B1&B2	*S_Button1* and *S_Button2* were TRUE when the FB was activated. *Ready* = TRUE *Error* = TRUE *S_TwoHandOut* = FALSE

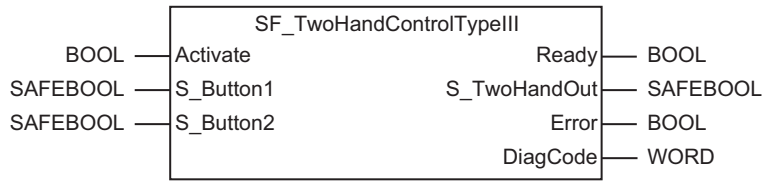
FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *Error* = FALSE *S_TwoHandOut* = FALSE
8000	32768	Buttons Actuated	Both buttons were operated correctly. The safety-related output is active. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = TRUE
8001	32769	Init	The FB is active but it is in the Init state. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE
8004	32772	Buttons Released	Neither of the buttons is being operated. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8005	32773	Button 1 Actuated	Only *Button1* is being operated. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE
8006	32774	Button 2 Actuated	Only *Button2* is being operated. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE
8007	32775	Button 2 Released	The safety output was enabled and then disabled again. After the safety output was disabled, *S_Button1* and *S_Button2* did not both change to FALSE. In this state, *S_Button1* is TRUE and *S_Button2* is FALSE after the safety output is disabled. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE
8008	32776	Button 1 Released	The safety output was enabled and then disabled again. After the safety output was disabled, *S_Button1* and *S_Button2* did not both change to FALSE. In this state, *S_Button1* is FALSE and *S_Button2* is TRUE after the safety output is disabled. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE
8009	32777	Locked Off	The safety output was enabled and then disabled again. After the safety output was disabled, *S_Button1* and *S_Button2* did not both change to FALSE. In this state, *S_Button1* is TRUE and *S_Button2* is TRUE after the safety output is disabled. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE
8019	32793	Locked On	The button operation was not correct. The FB is waiting for both buttons to be released. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE

SF_TwoHandControlTypeIII

This safety FB provides a type III, two-hand control function as defined in ISO 13851 (EN 574).

Instruction	Name	FB/FUN	Graphic expression
SF_TwoHandControlTypeIII	Two-Hand Control Type III	FB	

Variables

Input Variables

Variable	Data type	Valid range	Default	Description
Activate	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Input Variables* on page 4-2.
S_Button1	SAFEBOOL	TRUE or FALSE	FALSE	A variable. It is the button 1 input (category 3 or 4: for two antivalent contacts). FALSE: Button 1 is OFF. TRUE: Button 1 is ON.
S_Button2	SAFEBOOL	TRUE or FALSE	FALSE	A variable. It is the button 2 input (category 3 or 4: for two antivalent contacts). FALSE: Button 2 is OFF. TRUE: Button 2 is ON.

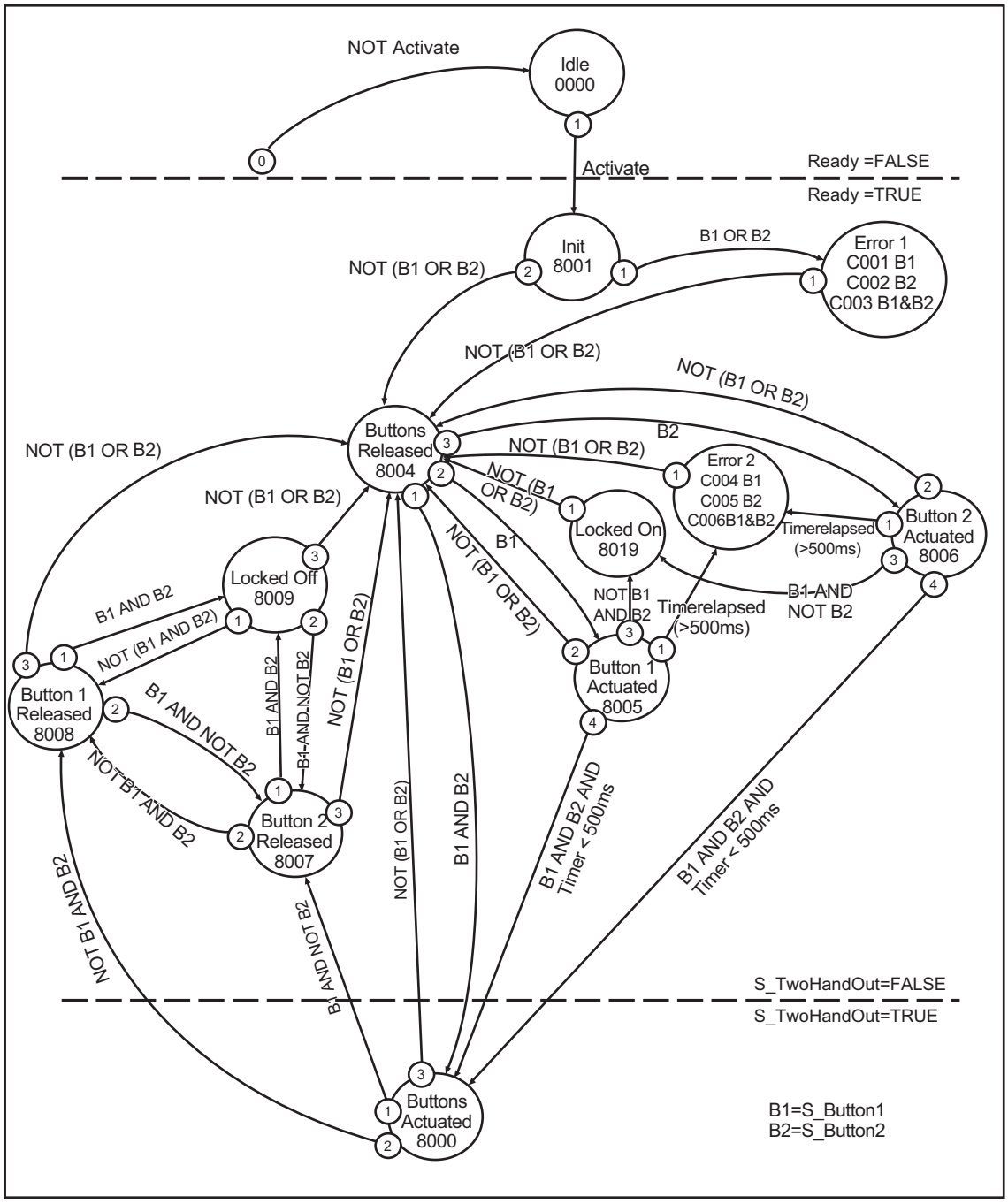
Output Variables

Variable	Data type	Valid range	Default	Description
Ready	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
S_TwoHandOut	SAFEBOOL	TRUE or FALSE	FALSE	The safety output signal. FALSE: A button is not being operated or correct two-hand operation was not performed. TRUE: The *S_Button1* and *S_Button2* inputs changed to TRUE within 500 ms and there is no error. Correct two-hand operation was performed.
Error	BOOL	TRUE or FALSE	FALSE	Refer to *Safety FB Common Output Variables* on page 4-4.
DiagCode	WORD	Depends on state code.	16#0000	Refer to *Safety FB Common Output Variables* on page 4-4.

Function

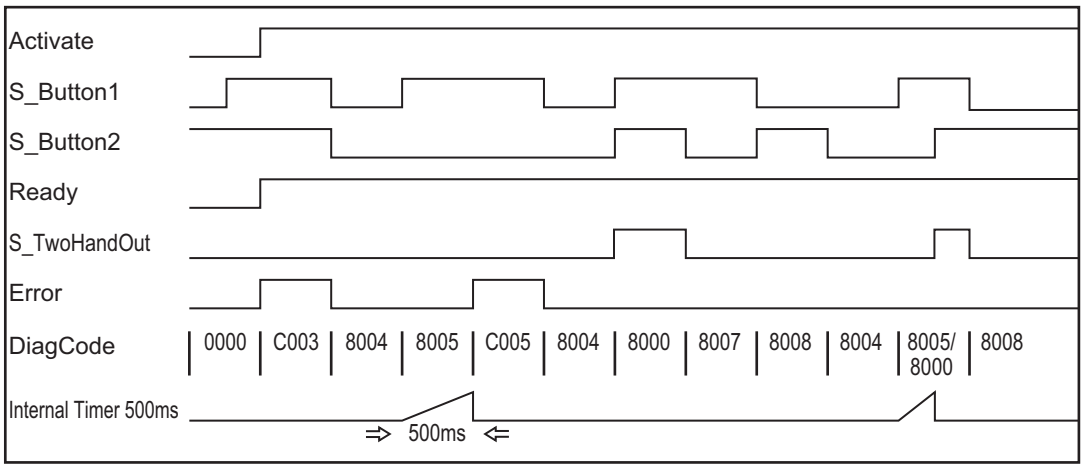
- This FB provides a type III, two-hand control function as defined in ISO 13851 (EN 574). If *S_Button1* and *S_Button2* are set to TRUE in the correct order within 500 ms, the *S_TwoHandOut* output is also set to TRUE.
- This FB also controls releasing of both buttons before it sets the *S_TwoHandOut* output to TRUE again.

State Transition Diagram



Note Transitions to the Idle state from any other state are not shown for when *Activate* changes to FALSE. However, the transition to the Idle state has the highest priority (0).

Timing Charts



Instruction Execution Errors

Error Detected

After the FB is activated, it detects buttons that are already set to TRUE as illegal input settings that result in errors. This FB detects if the input signal time difference exceeds 500 ms.

Operation for Errors

- When an error occurs, the *S_TwoHandOut* output is set to FALSE and the safe state is maintained.
- When both buttons are released (i.e., set to FALSE), the error status is reset.

FB-specific Error Codes

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
C001	49153	Error 1 B1	*S_Button1* was TRUE when the FB was activated. *Ready* = TRUE *Error* = TRUE *S_TwoHandOut* = FALSE
C002	49154	Error 1 B2	*S_Button2* was TRUE when the FB was activated. *Ready* = TRUE *Error* = TRUE *S_TwoHandOut* = FALSE
C003	49155	Error 1 B1&B2	*S_Button1* and *S_Button2* were TRUE when the FB was activated. *Ready* = TRUE *Error* = TRUE *S_TwoHandOut* = FALSE
C004	49156	Error 2 B1	After 500 ms in state 8005, *S_Button1* was FALSE and *S_Button2* was TRUE. *Ready* = TRUE *Error* = TRUE *S_TwoHandOut* = FALSE
C005	49157	Error 2 B2	After 500 ms in state 8005, *S_Button1* was TRUE and *S_Button2* was FALSE. *Ready* = TRUE *Error* = TRUE *S_TwoHandOut* = FALSE
C006	49158	Error 2 B1&B2	After 500 ms in state 8005 or 8006, *S_Button1* was TRUE and *S_Button2* was TRUE. This state is possible only when the *S_Button1* and *S_Button2* input status change from different status to the same status (both TRUE) when the timer expires (500 ms) in the same cycle. *Ready* = TRUE *Error* = TRUE *S_TwoHandOut* = FALSE

● FB-specific State Codes (No Error)

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
0000	0	Idle	The FB is disabled (default). *Ready* = FALSE *Error* = FALSE *S_TwoHandOut* = FALSE
8000	32768	Buttons Actuated	Both buttons were operated correctly. The safety output is active. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = TRUE
8001	32769	Init	The FB is active but it is in the Init state. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE
8004	32772	Buttons Released	Neither of the buttons is being operated. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE
8005	32773	Button 1 Actuated	Only *Button1* is being operated. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE
8006	32774	Button 2 Actuated	Only *Button2* is being operated. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE
8007	32775	Button 2 Released	The safety output was enabled and then disabled again. After the safety output was disabled, *S_Button1* and *S_Button2* did not both change to FALSE. In this state, *S_Button1* is TRUE and *S_Button2* is FALSE after the safety output is disabled. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE
8008	32776	Button 1 Released	The safety output was enabled and then disabled again. After the safety output was disabled, *S_Button1* and *S_Button2* did not both change to FALSE. In this state, *S_Button1* is FALSE and *S_Button2* is TRUE after the safety output is disabled. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE

DiagCode (hexadecimal)	DiagCode (decimal)	Status name	Status description and output results
8009	32777	Locked Off	The safety output was enabled and then disabled again. After the safety output was disabled, *S_Button1* and *S_Button2* did not both change to FALSE. In this state, *S_Button1* is TRUE and *S_Button2* is TRUE after the safety output is disabled. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE
8019	32793	Locked On	The button operation was not correct. The FB is waiting for both buttons to be released. *Ready* = TRUE *Error* = FALSE *S_TwoHandOut* = FALSE

I

Index

Index

M

N

O

P

R

S

T

U

W

X

OMRON Corporation **Industrial Automation Company**
Kyoto, JAPAN
Contact: www.ia.omron.com

Regional Headquarters
OMRON EUROPE B.V.
Wegalaan 67-69, 2132 JD Hoofddorp
The Netherlands
Tel: (31)2356-81-300/Fax: (31)2356-81-388

OMRON ELECTRONICS LLC
2895 Greenspoint Parkway, Suite 200
Hoffman Estates, IL 60169 U.S.A.
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

OMRON ASIA PACIFIC PTE. LTD.
No. 438A Alexandra Road # 05-05/08 (Lobby 2),
Alexandra Technopark,
Singapore 119967
Tel: (65) 6835-3011/Fax: (65) 6835-2711

OMRON (CHINA) CO., LTD.
Room 2211, Bank of China Tower,
200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-5037-2222/Fax: (86) 21-5037-2200

Authorized Distributor:

Cat. No. Z931-E1-05 0719