PLCopen®
for efficiency in automation

PLCopen - Technical Committee 5 – Safety Software

Technical Specification

Part 1: Concepts and Function Blocks

Version 2.10 – Official Release

DISCLAIMER OF WARRANTIES

THE NAME 'PLCOPEN®' IS A REGISTERED TRADE MARK AND TOGETHER WITH THE PLCOPEN LOGOS OWNED BY THE ASSOCIATION PLCOPEN.

THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS AND MAY BE SUBJECT TO FUTURE ADDITIONS, MODIFICATIONS OR CORRECTIONS. PLCOPEN HEREBY DISCLAIMS ALL WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR SUITABILITY FOR A PARTICULAR PURPOSE, FOR THIS DOCUMENT. UNDER NO CIRCUMSTANCES WILL PLCOPEN BE RESPONSIBLE FOR ANY LOSS OR DAMAGE ARISING OR RESULTING FROM ANY DEFECT, ERROR, OR OMISSION IN THIS DOCUMENT OR FROM ANY USE OF OR RELIANCE ON THIS DOCUMENT.

Date: November 7, 2023

Total number of pages: 194

Concepts and Function Blocks for Safety Functions

The following paper is a document created within the PLCopen Technical Committee 5 – Safety Software. It is an update of version 2.01 as published in February 2020 and is merged with the corrigendum of March 2023, which contained feedback to (minor) errors in the document.
It summarizes the results of the PLCopen Technical Committee meetings, containing specific contributions for this version of the following persons of PLCopen member companies:

Name	*Company*
Abubakar Sanaullah	ABB AG
Alexandros Skiadas	ABB AG
Thomas Riess	Bosch Rexroth
Ulf Schünemann	Codesys
Harry Koop	Phoenix Contact Software
Eelco van der Wal	PLCopen
Roeland Hagesteijn	PLCopen

Change Status List:

Version Number	Date	Change Comment
V 2.0	July 3, 2018	Publication of V20 after no feedback to be included
V2.0A	October 2, 2019	As result of the decisions made for the corrigendum during the webmeeting
V2.0B	October 29, 2019	Additional feedback items discussed at webmeeting Oct. 24 and published intern.
V2.01	February 25, 2020	Released version since no additional comments received
V 2.01a	January 25, 2023	As result of the webmeeting on that date
V 2.02	March 16, 2023	As a result of the webmeeting on that day. All accepted changes included
V 2.03	July 10, 2023	Basis for the inclusion of the improved timing diagram
V 2.04	Aug. 21, 2023	As a result of the webmeeting
V 2.05	Aug. 22, 2023	Including new timing diagrams
V 2.06	Oct. 17, 2023	Last changes done. Final version before publication?
V 2.10	Nov. 7, 2023	Official Release

Table of Contents

Table of Figures

1 Introduction

The independent association PLCopen, together with its members and external safety-related organizations, has defined safety-related aspects within the IEC 61131-3 development environments. With this, the safety aspects can be transferred to a software tool, which is integrated into the software development tools. This combination helps developers to integrate safety-related functionality into their systems right from the beginning of the development cycle. Also, it contributes to the overall understanding of safety aspects, as well as certification and approval from independent safety-related organizations.
This document mainly focuses on machine controls and is aimed at both:

a) Suppliers of programmable safety controls
b) Users of programmable safety controls

With this addition, PLCopen merged three environments on one development platform: Logic, Motion, and Safety. This is shown in figure 1.

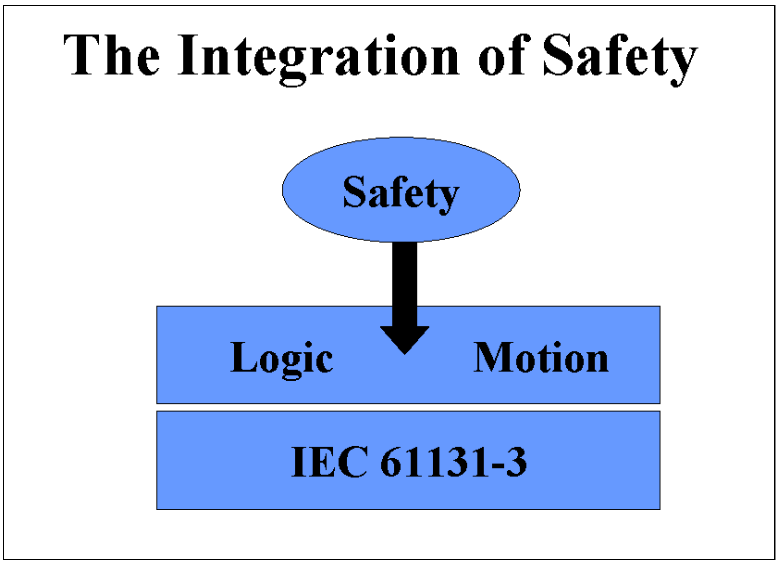


Figure 1: Merging three environments on one platform.

1.1 The Rationale of a New Safety Standard

Machine builders are faced with a large set of safety-related standards. This makes it expensive and, in some cases, unfeasible for machine builders to understand them all fully. Yet in the end they are still responsible for their products and related safety aspects. This risk situation is not very healthy, especially since legislation imposes greater constraints on the equipment suppliers. And their liability increases.
Nowadays there is often a clear separation between the safety-related part and the functional application part. This separation can be made be using different systems for the environments, different tools, and even different people can be involved. This separation often results in the safety aspects being included at the end, and not integrated into the whole system philosophy from the beginning, and often with only limited tests performed. This clearly does not contribute to the overall safety aspects.
Also, the on-going technological innovation now provides safety-approved digital communication buses. This supports the trend away from hard-wired systems towards software-oriented solutions. A parallel can be drawn with the movement away from hard-wired relay logic towards programmable logic controllers, PLCs. Such a trend, of course, involves a change in the mindset. This type of change requires time, widespread support from the industry, support from educational institutes as well as from certification bodies.
In addition, governmental requirements add to the complexity. For instance, the US-based FDA, Food and Drugs Administration, has set strict regulations that must be complied with. Non-compliance can result in heavy financial penalties, again weakening the sustainability of the organization.

The common basic requirements of a safety application for machine builders within all applicable safety standards are:

- Distinction between safety and non-safety functionalities
- Use of applicable programming languages and language subsets
- Use of validated software blocks
- Use of applicable programming guidelines

- Use of recognized error-reducing measures for the lifecycle of the safety-related software

For users, the effort to fulfill these high requirements should be reduced. This can be done using standardized solutions, which enable typical functionalities to be implemented easily. The standardization of function blocks and integration and support from software tools enables programmers to integrate safety in their applications from the beginning, without adversely affecting their functions and performance, and without adding costs.
To achieve this, PLCopen Committees are working on two levels:

1. **Standardization in the look and feel of safety function blocks**
2. **Integration of standard procedures in the development environment**

1: Standardization in the Look and Feel of Safety Function Blocks
In order to help developers to use safety-related functionalities, the comfort zone of users must be improved, thus making it easier to accept this way of working. This can be done by standardizing the look and feel of the safety function blocks. In this way the safety functionality can be better recognized and used independently of the applicable system. Re-training is not necessary and the tendency to create dedicated safety functionality is reduced.
In addition, this assists the certification bodies. Specifying and checking the safety software becomes much easier, and therefore quicker, less risky, and less costly.
Providing function blocks at a higher level makes them less dependent on the underlying hardware architecture. Architectures such as hard-wired systems, systems containing safe input and output modules, and network-based systems can be supported with the same function blocks. With this higher-level solution the implementation details can be hidden from users, making the implementation of safety-related software much easier and less costly. This also improves the comfort zone of users.

2: Integration of Standard Procedures
Once the functionalities have been presented in function blocks, the next stage is to determine how to combine them into safety-related programs. At this level the software tool should help the user as much as possible. For this, a new BOOLEAN data type is introduced that is applicable within the safety-related environment and provides a distinction between safety-related and non-safety-related Boolean variables. This provides the basis for the development tool to identify safety-critical program parts, and guide the user with permissible connections, while preventing incorrect connections. In this way, support can be implemented for the different levels of the various safety standards.
This is combined with a reduction in the functions of the programming languages. In addition, the Function Block Diagram and Ladder Diagram graphical languages are preferred, thus creating program parts that are easier to create and check. This represents a major contribution to the acceptance and use of safety-related functions, thus eliminating several obstacles as they now exist, and are described above, especially for the machine building industry.

1.2 Objectives

The following objectives were identified and met within this Technical Committee:

- Definition of a standard function block (FB) library for standard safety-related functionality.
- Combining these FBs with an application program requires an environment that is suitable for safety-related applications. Requirements and restrictions for such an environment are partly dealt with in this standard.
- Accepted concepts and functions by potential certification bodies, providing the basis for certifiable FBs (as objective for Version 1.0).
- Providing an easy-to-use interface to the safety functionality.
- Providing a common basis, terminology, and references.
- Related to existing safety standards.
- Providing a "style guide" for additional/future FBs
- Providing User Guidelines and examples.
- Application program should be reusable across platforms.
- The primary focus of this Technical Committee is safety in machinery.
- To include other areas beyond the machine building industry, further additions are expected. These additions can be dealt with in future additions to this document.
- This specification shall be seen as an open framework without hardware dependencies. It provides openness for implementation on different platforms. The actual implementation of the function blocks themselves is outside the scope of this standard.
- The programming of "safety-related" and "non-safety-related" logic may be possible in the same context.

Based on these objectives, the PLCopen Technical Committee 5 – Safety produced this specification to meet the basic safety requirements. This specification includes:

- Representation of the software architecture.
- Definition of the programming languages.
- Presentation of safety-related data types.
- Definition of language subsets.
- Definition of user levels for easy programming and error prevention.
- Error handling and diagnostic concept.
- Definition of a generic safety-related function block.
- The definition of a set of 22 safety-related function blocks.
- The definition of a PLCopen compliance procedure combined with the use of the PLCopen Safety logo.

This document basically consists of three parts:

1. Reduction in programming languages and functions, to enable safety-related application programs to be created.
2. General rules for safety-related function blocks.
3. The definition of a set of function blocks with safety-related functions.

1.3 Certification

This document provides guidelines, style guides, and basic specifications of function blocks for implementation and use in safety-related environments. The certification bodies confirm by reviewing resulting in a statement to PLCopen that this document, starting with Version 1.0, meets the relevant aspects of IEC 61508 and the related standards and can be used as a part of a specific safety requirement specification. By using the FBs together with the general aspects, the certification procedure of the application becomes much easier and faster. This also applies to the supplier of the software environment regarding the implementation of this specification. However, this document or a PLCopen certificate does not guarantee that the implementation meets the requirements of the safety standards. Therefore, the implementation of the FBs, or their appropriate use, is the responsibility of the supplier and/or user, including safety certification.

In order to meet the requirements as defined, different kinds of testing and certification are applicable:

1. Testing and certification of the software tool, often part of the control supplier.
2. Testing and certification/conformity of the safety application as programmed by the user.

Ad 1: Testing and certification of the software tool, often part of the control supplier
The development environment, including the safety-related function blocks, must be certified by the other relevant bodies. To become certified, certain regulations such as those described in IEC 61508 are applicable. These requirements are beyond the scope of this document.

Ad 2: Testing and certification/conformity of the safety application as programmed by the user.
Within an application, certification includes the safety-related software combined with the infrastructure, such as sensors, switches and actuators, connection schemes, etc. Certification or approvals for these environments are beyond the scope of this document and must be dealt with by external dedicated organizations.

The use of the PLCopen logo does not give any guarantees as to compliance with or fulfillment of criteria. The use of the logo simply indicates the inclusion of the concepts and guidelines as described in this document, within the relevant software environment, and the availability of this information in more detail in the relevant section of the PLCopen website: www.PLCopen.org .

1.4 Major Changes in this version

This document Version 2.0 is based on the original Version 1.0 document as published on January 31, 2006.
It incorporates the original Part 3, especially the section on diagnostics and the additional 5 function blocks.
In addition, the Structured Text language ST is added, as well as additional datatypes and functionalities.
All the original function blocks have been updated w.r.t. diagnostic codes, the outputs safety demand and reset requested, and the reset functionality has been extended to trailing edge via the definition of a new function block.
Also, there were 3 motion related function blocks removed and added to a separate document on SafeMotion.
In October 2019 the feedback items as listed in the Corrigendum were merged with V2.0 to create V2.01 on February 25, 2020. Further feedback items resulted in a new corrigendum, which, after multiple meetings, resulted in Version 2.1.

2 General

2.1 Scope

This paper enables conformance with the relevant software-related requirements as specified in IEC 61508 and other basic standards listed in chapter 2.3. As such it provides a basis for the software safety function requirements specification for safety-related function blocks for the implementer and provides guidance in the software design and coding phases for both the developer/implementer of the FB's and the user of the FB's. This function requirements specification is suitable for applications with required safety integrity levels of SIL 1, SIL 2 and SIL 3. SIL 3 is the highest SIL required for safety of machinery.

The IEC 61508 safety standard includes the description of a safety lifecycle. This contains 16 phases in total, starting with "1. Concept" and ending with "16: Decommissioning or disposal".
This PLCopen document contributes to IEC 61508 "Phase 9: Realisation; Software safety lifecycle 9.1.1; Safety function requirements specification".

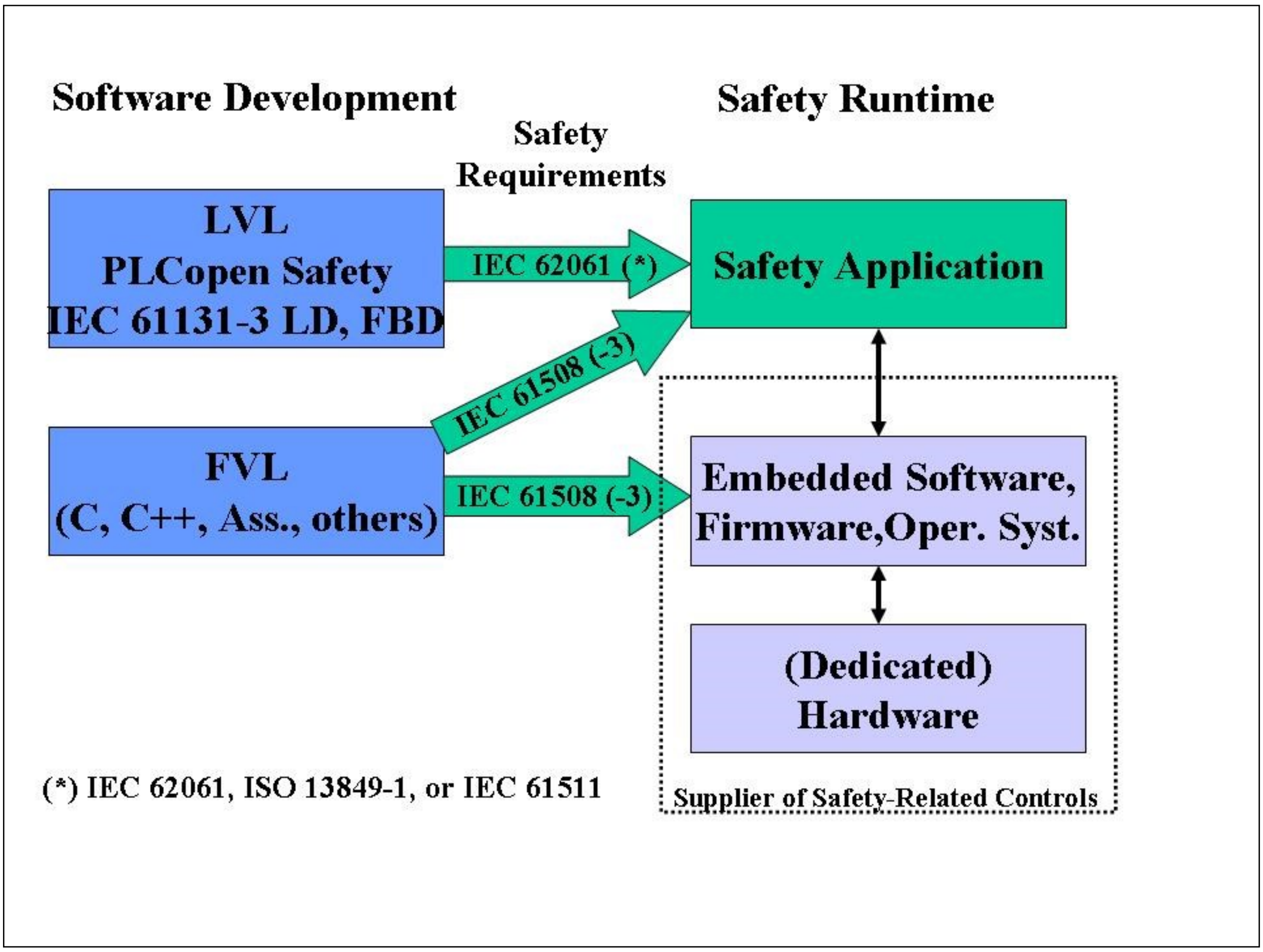


Figure 2: Focus of the work.

The relationship between the different standards, the development phases, and the runtime is shown in "Figure 2: Focus of the work". On the left side are the development environments for two levels of software:

1. The embedded software, firmware, or operating system, which must comply with the regulations of IEC 61508, especially Part 3. Languages used here can include C, C++, assembler, or others. These are Full Variability Languages (FVL): application-independent languages used by component suppliers for the implementation of (safety) firmware, operating systems, or development tools. Rarely used for the safety application itself.
2. The safety application software. If implemented with C, C++, assembler, or others, it is necessary to comply with IEC 61508 as above. They are again based on Full Variability Languages.
   If implemented according to this PLCopen specification, including the reductions in programming languages, instructions, and certified function blocks, the standards for machinery sector, i.e., IEC 62061 and ISO 13849-1, must be observed by the user at the targeted industries. This simplifies software development and approval dramatically. In this case they can be referred to as Limited Variability Languages (LVL). They are aimed at users to create their safety application function blocks. The languages typically used are Ladder Diagram and Function Block Diagram.

The function blocks specified here are not to be treated as a "subsystem element" as defined by IEC 62061, but as IEC 61131-3 function blocks. The IEC 62061 definition of a function block differs from that used in IEC 61131-3 in the sense that it can include hardware, providing safety subsystem functionality.

2.2 Terms and Definitions

AOPD	Active opto-electronic protective device
Basic Level	Programming level aimed at safety-application programmers using the certified (or validated) function blocks.
Categories/Cat.	According to EN 954, discrete level for specifying the safety integrity requirements of the safety functions to be allocated to the safety-related systems.
EDM	External device monitoring signal, which reflects the state transition of an actuator.
ESPE	Electro-sensitive protective equipment
Extended Level	Programming level which extends the basic level with the ability to define custom extensions to the specified set of function blocks as well as creating application programs.
FBD, LD, SFC, ST, IL	Programming Languages according to IEC 61131-3: FBD = Function Block Diagram, LD = Ladder Diagram, SFC = Sequential Function Chart, ST = Structured Text, IL = Instruction List
Function Block (FB)	According to IEC 61131-3, instance of a function block type, where a function block type is a programmable controller programming language element consisting of: 1) The definition of a data structure partitioned into input, output, and internal variables. 2) A set of operations to be performed on the elements of the data structure when an instance of the function block type is invoked.
Functional application software	General part of the application software, which is not directly related to the safety aspects.
FVL	According to IEC 62061, Full Variability Language: type of language that provides the capability to implement a wide variety of functions and applications
LVL	According to IEC 62061, Limited Variability Language: type of language that provides the capability to combine predefined, application specific library functions to implement the safety requirements specifications
MC-related function	Function relating to motion control applications. To be considered in relation to the set of PLCopen standards "Function Blocks for Motion Control".
Muting	Muting is the intended suppression of the safety function. This is required, e.g., when transporting the material into the danger zone.
NC	Break contact. Normally-Closed contacts disconnect the circuit when the relay is activated; the circuit is connected when the relay is inactive.
NO	Make contact. Normally-Open contacts connect the circuit when the relay is activated; the circuit is disconnected when the relay is inactive.
OSSD	Output Signal Switching Device
Performance Level (PL)	According to ISO 13849-1, discrete level for specifying the safety integrity requirements of the safety functions to be allocated to the safety-related systems, where "PL e" has the highest level of safety integrity and "PL a" has the lowest.
PES	Programmable Electronic System (see IEC 61508)
PFD/PFH	According to IEC 61508-1, probability of failure to perform design function on demand (PFD)/probability of a dangerous failure per hour (PFH).
PLC	Programmable Logic Controller
POU	Program organization units ‘Program', ‘Function', and ‘Function Block', as defined in IEC 61131-3
Process control	Control signal from the functional application for process control.
SAFEBOOL	Data type to identify safety-related BOOLEAN signals. See 3.2 Safe Data Types
SAFExxxx	Data type to identify safety-related signals of type xxxx (like SAFEINT).
Safety	Freedom from unacceptable risk (IEC 61508-4: 3.1.8/ISO/IEC Guide 51 second edition (1997 draft)).
Safety application software	Part of application software used to implement safety-related control functions within a safety-related system.
Safety demand	Request to the safety-related function block to set the output signal to the Safe state (FALSE).
Safety Integrity Level, SIL	According to IEC 61508-4, discrete level for specifying the safety integrity requirements of the safety functions to be allocated to the E/E/PE safety-related systems.

System Level	Specific programming level aimed at the implementation of the (specified) function blocks by suppliers. This level is not explained further in this document.

2.3 Relation to Other Standards

The following standards are referenced by this specification:

- IEC 61508-3 (2010), Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 3: Software requirements
- ISO 13849-1 (2015), Safety of machinery - Safety-related parts of control systems - Part 1: General principles for design
- ISO 13849-2 (2012), Safety of machinery - Safety-related parts of control systems - Part 2: Validation
- IEC 62061 (2015), Safety of machinery - Functional safety of safety-related electrical, electronic, and programmable electronic control systems
- IEC 60204-1:2009, Safety of machinery - Electrical equipment of machines - Part 1: General requirements
- ISO 12100:2010, Safety of machinery - Basic concepts, general principles for design.
- ISO 13850:2015, Safety of machinery - Emergency stop - Principles for design
- IEC 61496-1:2012, Safety of machinery - Electro-sensitive protective equipment - Part 1: General requirements and tests
- IEC 61800-5-2 (2016), Adjustable speed electrical power drive systems - Part 5-2: Safety Requirements - Functional safety
- EN 574 (2008), Safety of machinery - Two-hand control devices - Functional aspects - Principles for design
- EN 1037 (2008), Safety of machinery - Prevention of unexpected start-up
- IEC 61131-3 (2013-02) Programming Languages, 3rd edition
- PLCopen Coding Guidelines (2016-04)

3 Model

3.1 Software Architectural Model

A software architectural model is provided to describe the typical location of the specified safety function blocks within a machinery control system. This model is as generic as possible, so that existing and upcoming safety control systems can be covered by this model. No safety control hardware architecture should be excluded by this software specification.

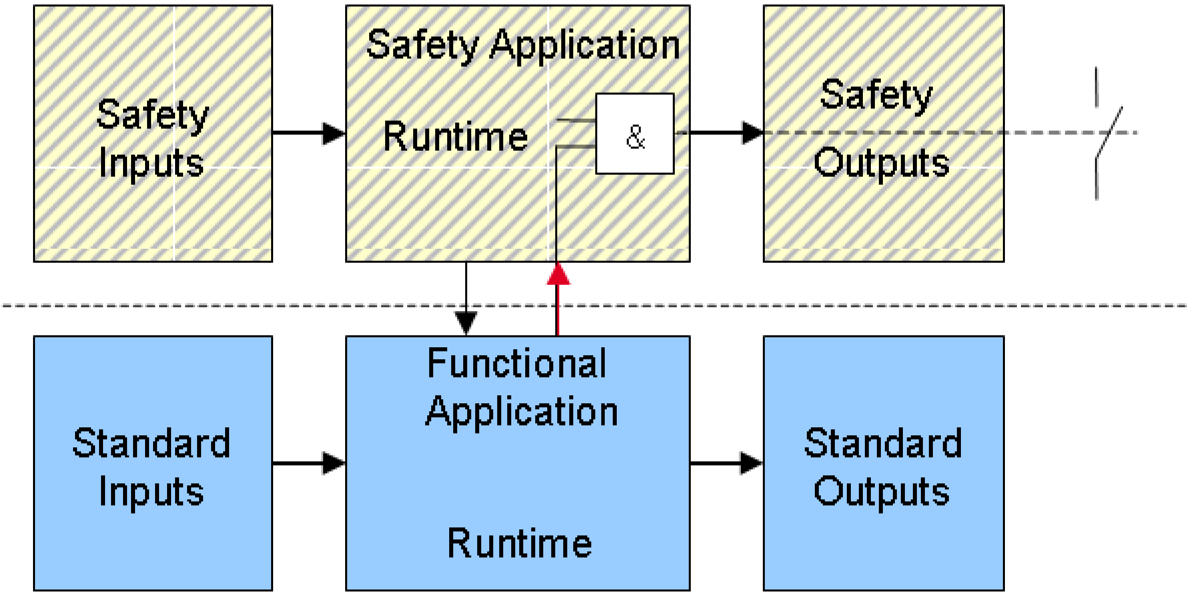


Figure 3: Architectural model

The proposed architectural model differentiates between the functional application part and the safety application part. This is often coupled to two levels of software engineering environments. The objective of PLCopen is to merge these two environments, e.g., a development environment for the functional part with an integrated safety part, including reductions in programming languages and functionality for the safety section.

The two applications could be executed on one device or there could be two or more separate devices which are more or less loosely coupled. The data exchange between the applications, represented by the dashed line, could be via networks, wired I/O or memory transfer within one device. Generally, an important requirement is that there is no undesirable interference from the functional application on the safety application.

On the left side of the model, two sets of inputs are identified, and on the right side two levels of outputs. In the middle, the two environments are shown separately, both coupled to their related inputs and outputs.
The permitted data exchange between the safety and the functional applications is shown in the middle.

- The functional application has read access to the safety inputs and global variables (as indicated by the left arrow).
- The non-safe signals can only be used in the safety application to control program flow and cannot be connected directly to the safe outputs (as indicated by the right arrow and the AND operator).

The same applies to the two sets of outputs.

Figure 4: Added Architectural shows on the left side the separation between the Safety and Functional Application. However, from the safety function view (right side), safety and standard variables effect the processing of the Safety FBs, even though the standard variables do not affect the safety at all since the safety application has to enable any safety related action.

To recognize the corresponding signals easier, the Input Signals get sorted in correspondence to the safety functions and get assigned directly to the safety application, irrespective if they belong to the functional or safety application, as shown on the right side in the figure above. Most of these standard signals (1) do not get manipulated by the Functional Application anyway. Some systems can address the Standard Inputs/Outputs directly from the Safety Application; others provide these signals through the Functional Application.
However, the separation to the different physical input/output blocks and the assignment to the functional or safety application still match with the basic architectural model as defined in figure on the left. The signals (2), processed by the Functional-

application and which do not belong to specific safety function such as reset, diagnosis, are simplified illustrated at the interface between the two applications.

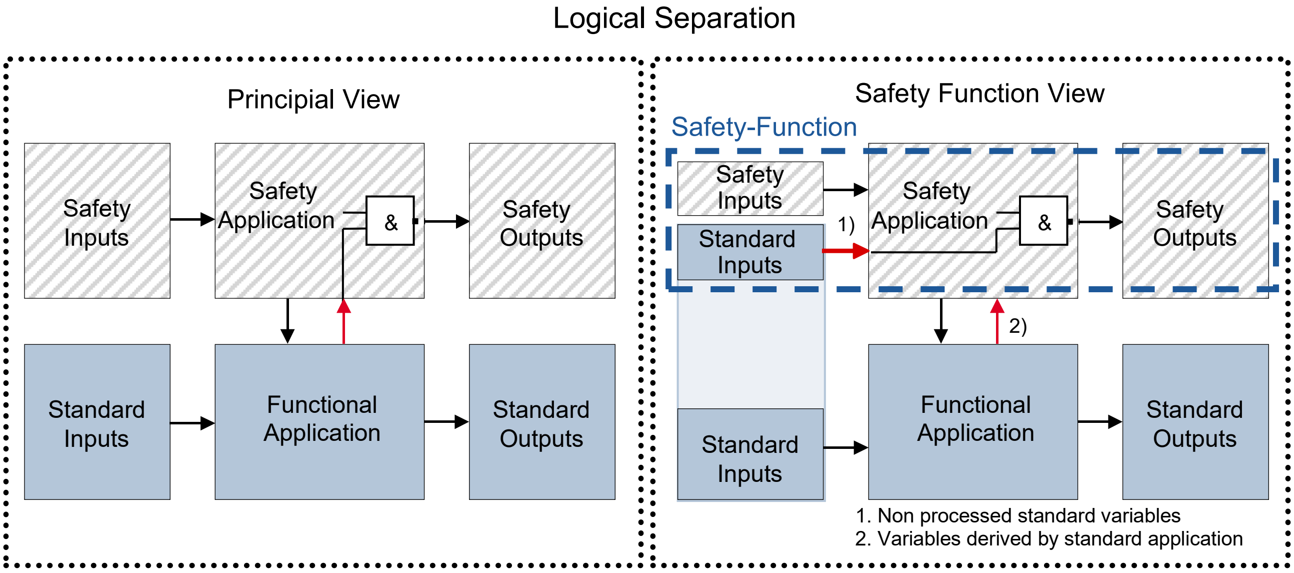


Figure 4: Added Architectural model.

A different representation is shown in Figure 5: Alternative view of the architectural model, where in the controller section the input monitoring (or processing) is done, then the logic follows, and then the output monitoring or processing.

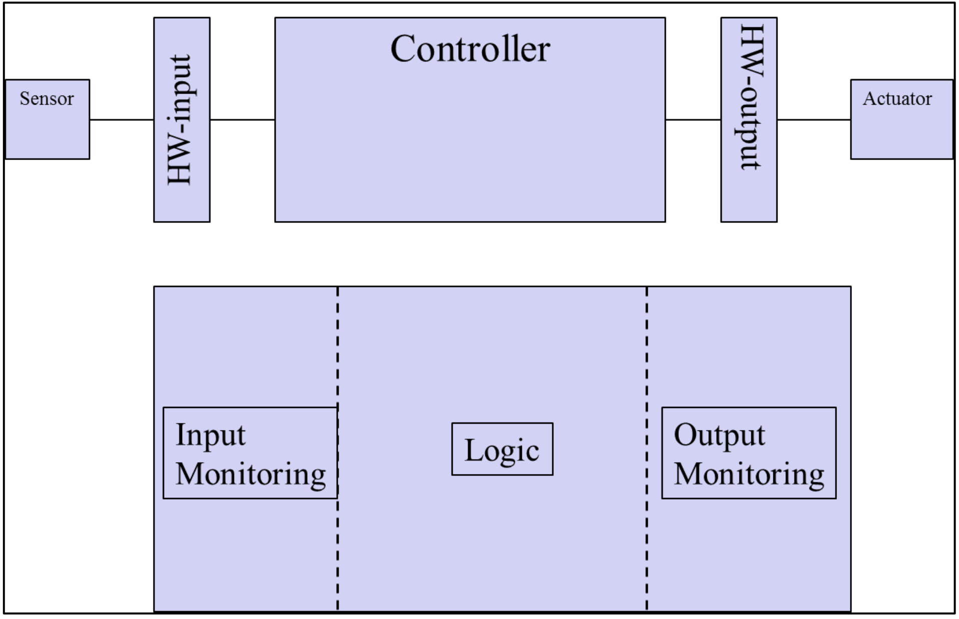


Figure 5: Alternative view of the architectural model

A more extensive overview of the model dedicated to the safety environment is shown in Figure 6: Layers in the architectural model. It consists of several levels within a safety application, i.e., between the safe inputs, the program with FBs, and the safe outputs. These levels are:

- Safety inputs (sensors)
- Input level
- Input processing level
- User interface level with function blocks
- Output processing level
- Output level
- Safety outputs

The safe inputs are made available to the software by the system. The details of this are outside the scope of this document. The same applies to the safe outputs. The SAFEBOOL data type is used to identify safe signals, including inputs and outputs within the software – the underlying technology is not part of this specification.

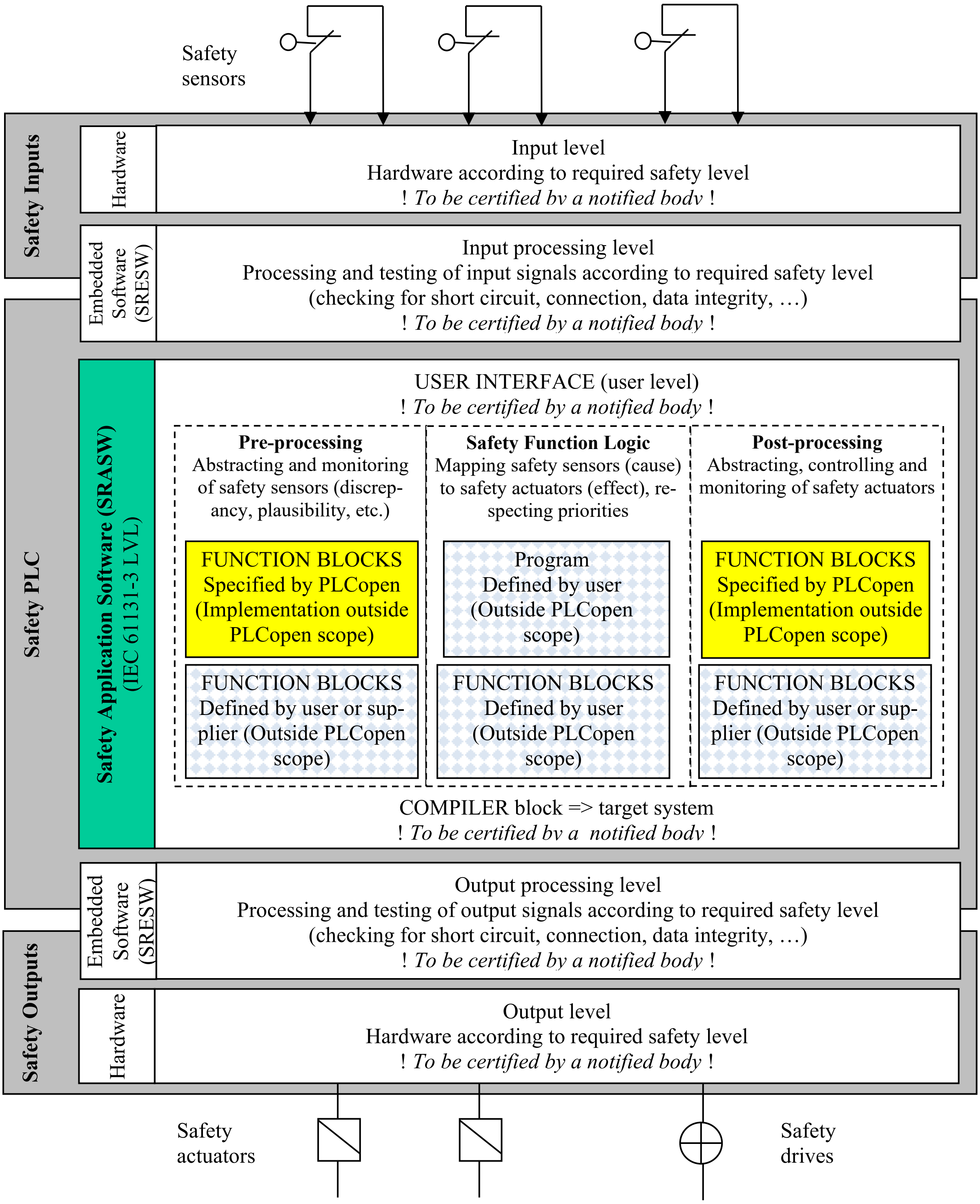

Safety sensors
Safety Inputs
Hardware
Input level
Hardware according to required safety level
! To be certified bv a notified bodv !
Embedded Software (SRESW)
Input processing level
Processing and testing of input signals according to required safety level
(checking for short circuit, connection, data integrity, …)
! To be certified by a notified body !
Safety PLC
Safety Application Software (SRASW) (IEC 61131-3 LVL)
USER INTERFACE (user level)
! To be certified by a notified body !
Pre-processing
Abstracting and monitoring of safety sensors (discrepancy, plausibility, etc.)
Safety Function Logic
Mapping safety sensors (cause) to safety actuators (effect), respecting priorities
Post-processing
Abstracting, controlling and monitoring of safety actuators
FUNCTION BLOCKS Specified by PLCopen (Implementation outside PLCopen scope)
Program Defined by user (Outside PLCopen scope)
FUNCTION BLOCKS Specified by PLCopen (Implementation outside PLCopen scope)
FUNCTION BLOCKS Defined by user or supplier (Outside PLCopen scope)
FUNCTION BLOCKS Defined by user (Outside PLCopen scope)
FUNCTION BLOCKS Defined by user or supplier (Outside PLCopen scope)
COMPILER block => target system
! To be certified bv a notified bodv !
Embedded Software (SRESW)
Output processing level
Processing and testing of output signals according to required safety level
(checking for short circuit, connection, data integrity, …)
! To be certified by a notified body !
Safety Outputs
Hardware
Output level
Hardware according to required safety level
! To be certified by a notified body !
Safety actuators
Safety drives

Figure 6: Layers in the architectural model

Notes:

1. The highlighted block in the drawing indicates the scope of this document. The surrounding functionalities are not part of this specification.
2. The number of inputs and outputs do not represent a real application.

The three stages (Pre-processing, safety-function logic, post-processing) in the architectural model of the application software can be implemented by IEC 61131-3 language elements at different levels of structuring, depending on the size and complexity of the application (see Figure 7: Structure of the safety application):

A. Smaller application may be implemented as a single program with access to the safe inputs/outputs.
   a. with all three stages implemented in one network (data flow between stages explicit as FBD connections), or
   b. with stages implemented in different networks (data flow between stages implicitly via local variables).

B. Larger application may be implemented by one program calling, for each stage, application-specific single-instance function blocks with access to the safe inputs/outputs (data flow between stages explicit as FBD connections), or

C. by a sequence of programs within the same task and with access to the safe inputs/outputs and implementing the different stages (data flow between stages implicitly via global variables).

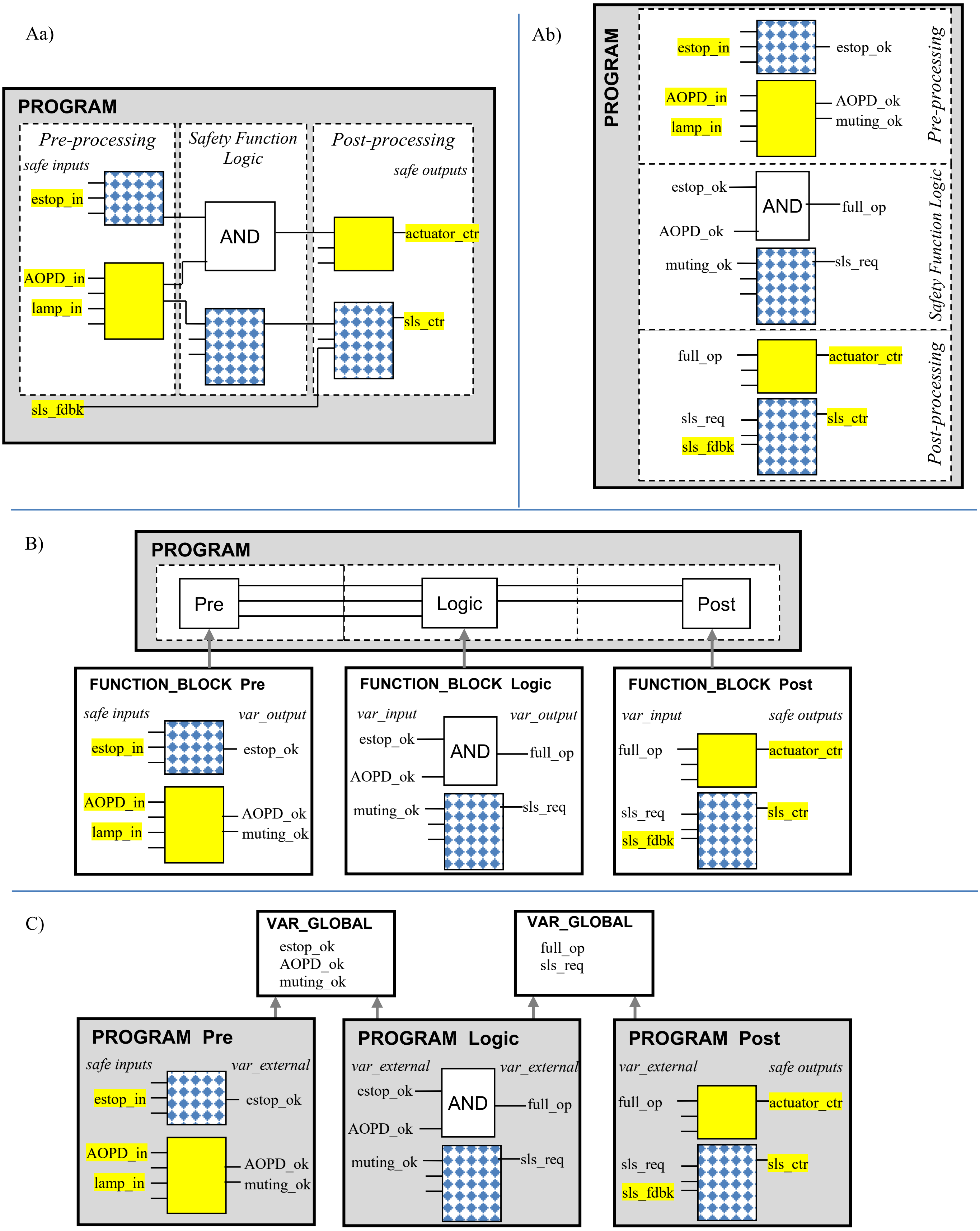


Figure 7: Structure of the safety application

3.2 Safe Data Types

In order to differentiate clearly between safety-relevant and standard signals, a new data type with the designation "SAFE" was defined. Thus, the programmer recognizes that the signals are safety-relevant and must be treated with special care. Furthermore, because of this designation the data links can be verified automatically to detect any impermissible links between standard signals and safety-relevant signals. Although the "SAFE" data type cannot guarantee that the signal status is safe (e.g., in the event of incorrectly wired periphery), it is, however, an organizational tool used to minimize errors in the application program. Additionally, when releasing the application program, the safety-relevant signals can be clearly recognized. This simplifies and shortens signal flow verification.
Safe data types are data types applicable within the safety-related environment. These data types shall be used to differentiate between safe signals and non-safe signals for ease of validation and certification purposes.

Possible means of supporting safety-related data types in programming environments could be:

- Different means of display/representation of safe data types
- Compiler support of safe data types

For instance, SAFEBOOL is a data type that is applicable within the safety-related environment and represents a higher safety integrity level. It differentiates between safety-related and non-safety-related variables. A SAFEBOOL acts as a BOOL within the system but can contain additional information (attributes) necessary for the safety status and level (could include categories/PL, SILs, PFD/PFH). Such information could be used to calculate the SIL with the programming tool.

The control system guarantees the Safety Integrity Level within the system limits. SAFExx variables are represented as "single-channel", regardless of the internal structure (which can be 1oo1, 1oo2D, 2oo2 or 2oo3). Therefore, such control systems, which execute FB's with SAFExx inputs and outputs, are to be certified, especially in respect of the generation of SAFExx signals.

Essentially there are (at least) two ways to get a SAFExx variable in the application level:

1. The data is provided as a safe data type by the devices, either by the devices themselves or by the operating system or firmware. This can include a safe network.
2. The data is provided by combining safety inputs in the application itself (such as two safe single-channel inputs).

The safe value for SAFEBOOL must be FALSE. Application designers must ensure that all SAFEBOOL variables result in safe behavior when set to FALSE. SAFEBOOL variables are set to FALSE on initialization and following any faults.
The default values (Safe Values) for the other datatypes SAFEREAL, SAFEINT, SAFEDURATION, SAFEBIT, SAFEDATE: are in accordance with IEC 61131-3 (2012) Table 10 – Default initial values, meaning in most cases binary 0.

3.3 General Recommendations and Constraints

- Program organization recommendation: The safety application program runs only as a single task. The functional application, which can be executed on a separate processor or device, can contain several tasks.
- The safety program shall not be interrupted by the functional application program.
- When the safety application cycle is started, all relevant input data representation is up-to-date and stable during the cycle.
- The safety-related outputs shall not be changed by the functional application alone.
- In the safety program it is recommended that certified function blocks, as defined in this specification, be used. The user can thus achieve a high level of error prevention.
- The safety function blocks shall be applicable in the FBD, LD and ST IEC 61131-3 languages, while the contents of the function blocks can be implemented in any programming language (e.g., IEC 61131-3 ST, C) or even in firmware or hardware. Therefore, the contents are not expected to be portable.
- Every POU/FB in the safety application has accessible information that contains the following: author, date of creation, date of release, version, version history, and functional description (including I/O parameters). This information is visible as a minimum during certification, program design, and program modification. Access to this information may vary depending on the type of use, e.g., can be part of the FB or can be referenced to another source like a web server.
- The software tool should provide support for header information in user defined POUs.
- The software tool should provide support to limit the complexity of the application configurable by the user. Examples for useful complexity measures will be given in Part 2 – User Guidelines

There shall be safety-versions named "SF_*fbname*" of standard FBs named "*fbname*" (as long as they are permitted for the programming level) differing only in the SAFE-qualification of outputs and input types: Outputs and certain inputs of type

BOOL (and more generally of any type *T*) are changed to SAFEBOOL (or to SAFE-*T*, resp., if the SAFE-qualification of *T* is supported). Namely:

- SF_TON/SF_TOF/SF_TP with Q: SAFEBOOL
- SF_SR, SF_RS with SET1, SET, Q, Q1: SAFEBOOL
- SF_CTU, SF_CTD, SF_CTUD with Q, QU, QD: SAFEBOOL and PV, CV: SAFEINT
- SF_R_TRIG and SF_F_TRIG with CLK, Q: SAFEBOOL

Note: Safety-related systems are based on "negative" logic. For instance, the physical emergency stop switch is normally closed, so a current flows through the circuit. If the switch is engaged, the contact opens, and so the current flow is stopped. ("Idle current" principle or "Ruhestrom-Prinzip" in the German language).

4 Reduction in the Development Environment

4.1 Definition of User Levels

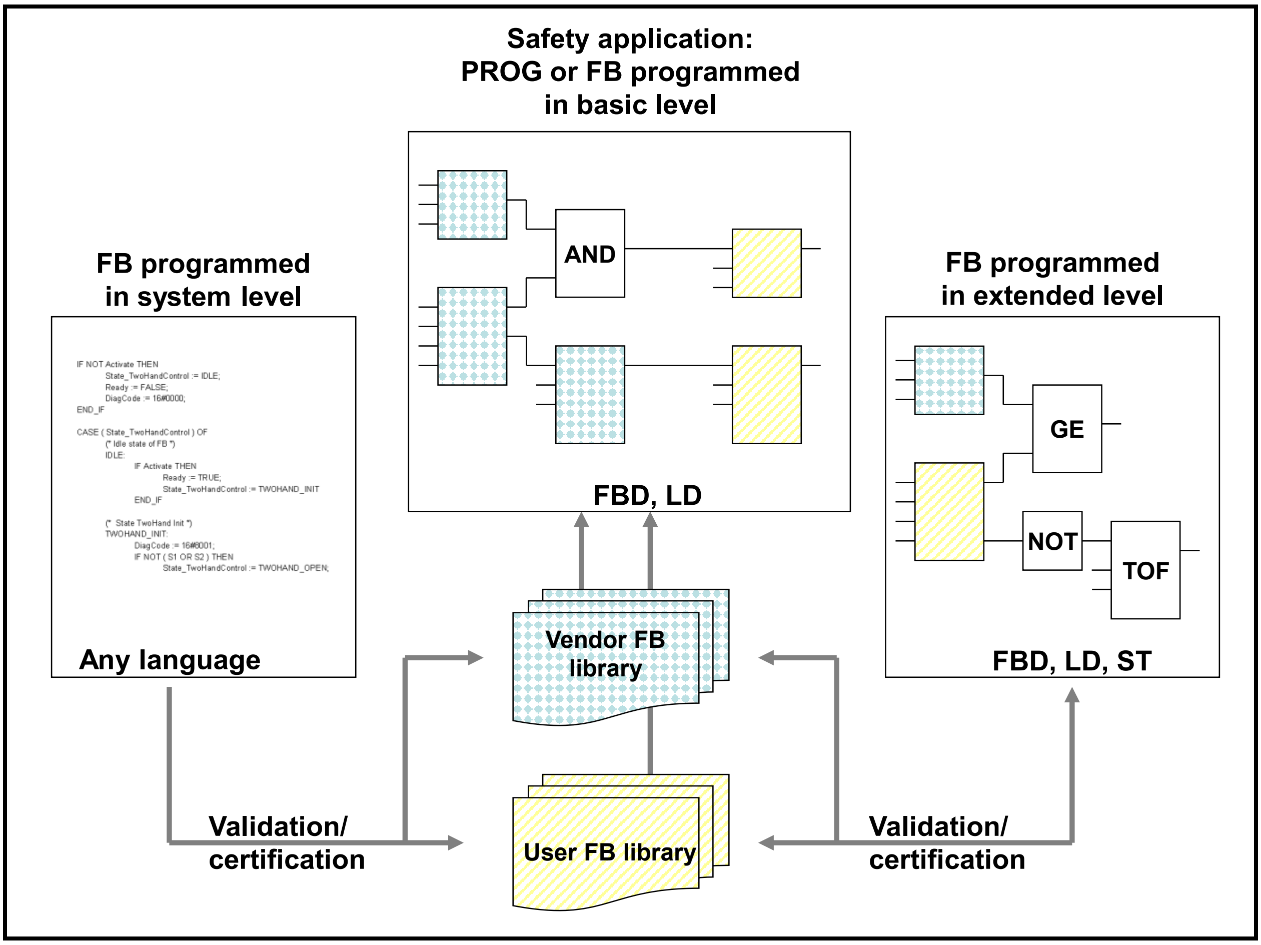
This specification differentiates between three levels:

Basic Level:
A fundamental approach is that the safety program only consists of certified function blocks (as well as user-derived function blocks which encapsulate basic level function blocks) that can be easily "wired" with one another in graphical form. If, in addition to this, the type of connection is limited, a view adapted to modern technology can be produced, which is similar to the discrete wiring of safety components. The programs have a clear structure and can be easily read. Furthermore, the release time of the program is significantly shortened, as it consists of blocks certified in advance.

Extended Level:
In the case of projects, for which the current status of certified function blocks is not sufficient, the user can create the required blocks (or even the program) in the Extended Level. For this, an extended command range is provided. However, the validation of the functionality for these blocks and programs can be considerably more complex and therefore more time-consuming since the programs underlie the whole verification process. If the blocks have been certified / validated, they can be used in the Basic Level together with the advantages described above.

System Level:
The System Level is provided for suppliers of safety controls. The System Level also enables, e.g., implementations in supplier-specific languages. However, the System Level is not part of the specification.

In any case, the different levels are integrated in the programming tool. Together with an access control they can be assigned to different user groups. The principle described above reduces the effort for the user significantly by simplifying the releasing process.



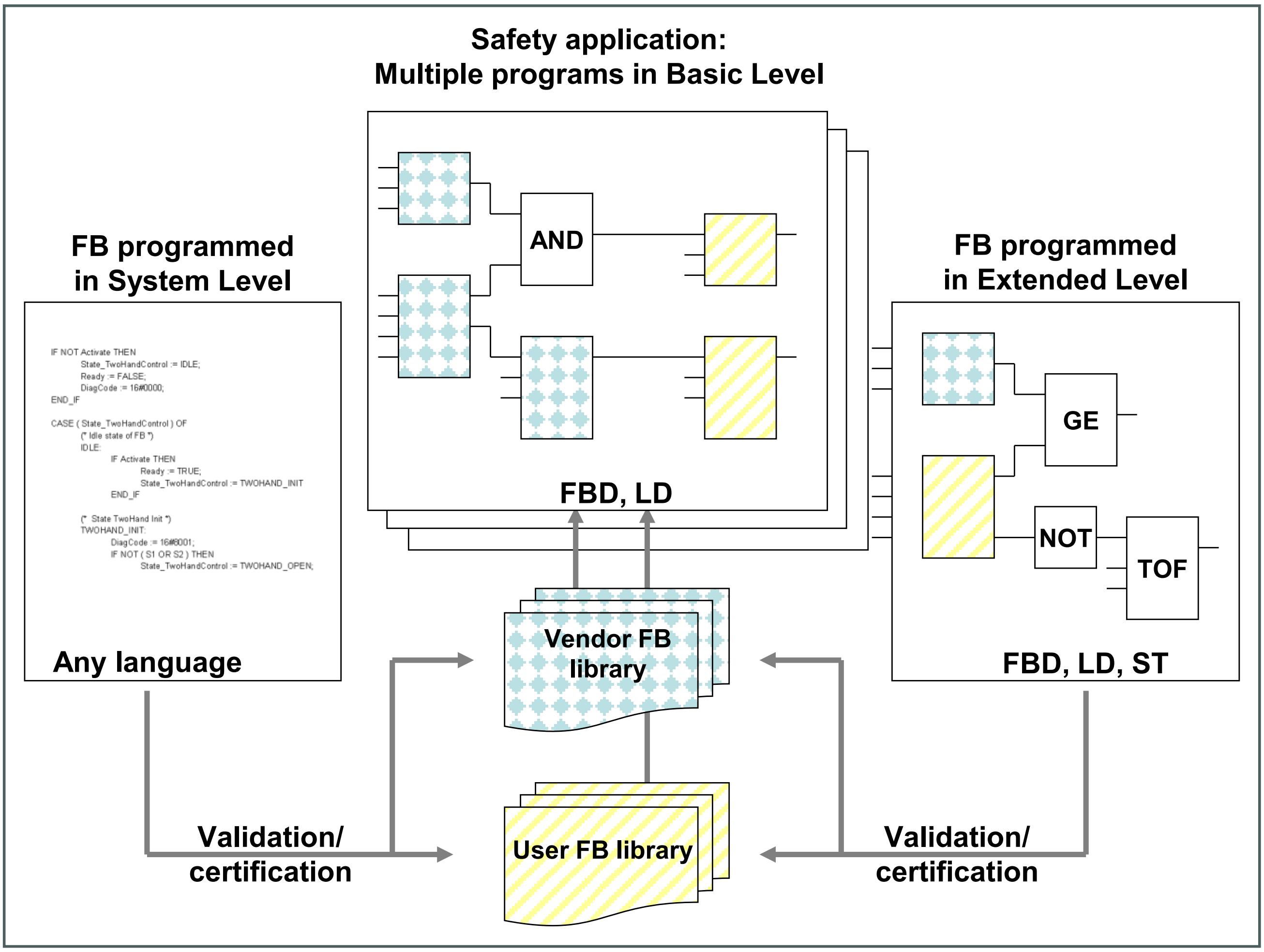
```
IF NOT Activate THEN
        State_TwoHandControl := IDLE;
        Ready := FALSE;
        DiagCode := 16#0000;
END_IF

CASE ( State_TwoHandControl ) OF
        (* Idle state of FB *)
        IDLE:
                IF Activate THEN
                        Ready := TRUE;
                        State_TwoHandControl := TWOHAND_INIT
                END_IF

        (*  State TwoHand Init *)
        TWOHAND_INIT:
                DiagCode := 16#8001;
                IF NOT ( S1 OR S2 ) THEN
                        State_TwoHandControl := TWOHAND_OPEN;
```

Figure 8: Recommended application scope of the three levels

Alternatively, one can divide the functionalities over several programs, like shown in the figure hereunder, what is in line with the multiple programs in Figure 7: Structure of the safety application. The communication between the different programs is done via global variables. All programs are in the same task and the sequence of the different programs must be able to be defined by the tool.



```
IF NOT Activate THEN
        State_TwoHandControl := IDLE;
        Ready := FALSE;
        DiagCode := 16#0000;
END_IF

CASE ( State_TwoHandControl ) OF
        (* Idle state of FB *)
        IDLE:
                IF Activate THEN
                        Ready := TRUE;
                        State_TwoHandControl := TWOHAND_INIT
                END_IF

        (*  State TwoHand Init *)
        TWOHAND_INIT:
                DiagCode := 16#8001;
                IF NOT ( S1 OR S2 ) THEN
                        State_TwoHandControl := TWOHAND_OPEN;
```

Figure 9: Alternative application scope of the three levels

4.2 Reduction in the Set of Programming Languages

IEC 61508, Part 7, defines a reduction in the preferred programming languages for the different SILs ("Highly Recommended", "Recommended" or "Not Recommended"). Based on this, the preferred languages within this specification are the Function Block Diagram (FBD) and Ladder Diagram (LD) graphical languages with a defined subset of the two. These graphical languages provide a clear overview of the safety program itself, and tool suppliers can implement a much better level of support and guidance for users. This forms the basis for simplified commissioning of the safety-related program. In addition, Structured Text (ST) is supported as textual language for usage in Extended Level. Instruction List (IL), and Sequential Function Chart (SFC) are not dealt with at this time, since higher lifecycle costs are anticipated. More specifically, the testing and validation of applications written in ST or IL is more complex and error-prone then applications written in graphical languages. This recommendation is specifically aimed at both the Basic Level and the Extended Level. No definitions in terms of languages, functions, and data types are provided here for the System Level (see IEC 61508, Part 7).

4.3 Reduction in Data Types and Declarations

The applicable datatypes are structured as shown in Figure 5 – Generic Datatypes of the 3rd edition of the IEC 61131-3 standard. This means that of the datatypes listed a safe equivalent can be defined, e.g., SAFEREAL, SAFEINT, SAFETIME, SAFEWORD. The light grey area is not applicable.

Generic data types	Generic data types	Groups of elementary data types
ANY		
ANY_DERIVED		
ANY_ELEMENTARY		
ANY_MAGNITUDE		
ANY_NUM		
ANY_REAL		REAL, LREAL
ANY_INT	ANY_UNSIGNED	USINT, UINT, UDINT, ULINT
	ANY_SIGNED	SINT, INT, DINT, LINT
ANY_DURATION		TIME, LTIME
ANY_BIT		BOOL, BYTE, WORD, DWORD, LWORD
ANY_CHARS		
ANY_STRING		STRING, WSTRING
ANY_CHAR		CHAR, WCHAR
ANY_DATE		DATE_AND_TIME, LDT, DATE, TIME_OF_DAY, LTOD

In the tables below, "X" indicates that the item is permitted, "-" indicates that it is not permitted.
(See IEC 61131-3; Table 10)

Description	**Basic Level**	**Extended Level**	**Comments**
ANY_REAL **ANY_SAFEREAL**	X	X	Basic Level: Arithmetic functions are not permitted. Extended Level: Arithmetic functions are permitted. Only valid values for REAL and SAFEREAL are allowed. This needs to be tested by the system and if not valid an exception is initiated, and the system goes in to the safe state.
ANY_INT; ANY_SAFEINT	X	X	Basic Level: Arithmetic functions are not permitted. Extended Level: Arithmetic functions are permitted. Note: Concerning overflow in SAFEINT operations: there should be measures included in the system to detect an overflow and handle it in an application specific way.
ANY_DURATION **ANY_SAFEDURATION**	X	X	Basic Level: Only as a constant FB input parameter and/or as outputs for diagnosis on the called FB. Extended Level: no restrictions
ANY_BIT **ANY_4.3**	X	X	Basic Level: Only as outputs for diagnosis on the called FB and as inputs for processing diagnostics codes from another FB. Extended Level: no restrictions
ANY_DATE	X	X	
ANY_SAFEDATE	-	X	

Note:
The SAFE-DATATYPEs are strongly recommended new data types for safety related signals. (For tools where these data types cannot be implemented, the use of standard data types is permitted. However, in that case, data type checking by the compiler is not possible. The user, or tool, is then responsible for ensuring safety and non-safety signals are not mixed up, which may lead to downgrading of the safety integrity level of safety functions.)

User-defined data types (See IEC 61131-3; Table 11, 12)

Description	**Basic Level**	**Extended Level**	**Comments**
Enumerated data types; **Data types with named values;** **Subrange data types;** **Directly derived data types;** **Reference types**	-	-	

Array data types	-	-	
Structured data type	X	X	To increase the readability. A struct variable (declaration in own POU, or struct parameter of an fb-instance, or struct element of a struct variable) can be used by accessing its elements using the "." notation (whitebox use), or by accessing the structure as a whole (blackbox use). Basic Level: only blackbox use allowed. This permits the passing of structured values between physical inputs and outputs and between the inputs and outputs of this Basic-level POU and of Extend-level FBs. Mixed use would complicate data flow analysis. Extended Level: mixed whitebox/blackbox use allowed. The whitebox case allows to define POUs that evaluate and/or compose structured contents themselves, or to create an entire set of variables to use by the declaration of one struct variable.
- **elements of elementary data types**	X	X	Restrictions on its elements' data types scale up to the structured data type. E.g., a STRUCT with TIME elements may only be used as constant input or for diagnostic output. All elements of a structured data types shall be uniformly: either all safety-related or not; no mixture of SAFE and non-SAFE types.
- **elements of STRUCT type**	X	X	No mixture of safety-related and non-safety-related struct elements.
- **elements of FB-type**	-	-	
- **elements with/without default initial value**	X	X	Priority over the element type's default initial value.
- **elements with AT**	-	-	The bit-layout of data in a structure and in the process-image is system level knowledge, not limited variability applicative programming.
Initialization using constant expression	X	X	
Struct variable declaration with/without element initialization	X	X	Priority over the initial value in the structured data type declaration. Necessary for CONSTANT struct variables.

Variable Declaration Keywords: (See IEC 61131-3; Table 19 for Functions, Table 40 (for FBs) and Table 47 (for Programs))

Description	**Basic Level**	**Extended Level**	**Comments**
VAR	X	X	Only via symbolic declaration. Do not declare local variables that are not used [CP24] All variables shall be initialized before being used [CP3]
VAR_TEMP	-	-	Do not use
VAR_INPUT, VAR_OUTPUT	X	X	Conform [CP17] - Each input parameter should be read at least once in the POU code - Each input parameter should not be written in the POU code - Each output parameter shall be written only once in the POU code - Each input/output parameter should be either read or written in the POU code
EN/ENO	-	-	Specified FB shall have at least one binary input (i.e., ACTIVATE) and one binary output (i.e., READY), so EN/ENO is not strictly required.

VAR_INPUT CONSTANT	X	X	Only CONSTANT values may be assigned to this input. Note: this is a proposed extension to IEC 61131-3 to support the programming tool to check if the input is connected to a CONSTANT. This is an option.
VAR_IN_OUT	-	-	
VAR_GLOBAL; **VAR_EXTERNAL** **(on FB Level)**			The use of global data may lead to adverse effects and could complicate the analysis of data flow (esp. with multiple instances of the same FB). The use of global variable is only justified in the following cases [CP18]: • Exchanging data with external devices (physical I/O, communication variables, …) • Exchanging data with the functional application Physical outputs shall be written once in every cycle [CP12] Do not declare global variables that are not used [CP24]
VAR_GLOBAL; **VAR_EXTERNAL** **(on program level within a single task)**	X	X	Restricted use of global data is possible. The use of global data should improve the analysis of data flow. The use of global variables should be limited in preference for local variables. [CP18] In the following case, the use of global variable is justified: • Exchanging data among PROGRAM instances (in the same TASK) […] • Exchanging data with the System: accessing system-defined variable to use execution environment specific feature, such as CurrentTime, ErrorStatus, etc. • Exchanging data with external devices (physical I/O, communication variables,..) • Exchanging data with the functional application Do not declare global variables that are not used [CP24] A global variable shall be written only by one POU [CP26] Physical outputs shall be written once in every cycle [CP12]
VAR_GLOBAL_CONSTANT	X	X	
VAR_ACCESS	-	-	
CONSTANT	X	X	
RETAIN; NON_RETAIN	-	-	Can lead to foreseeable misuse and different behavior between cold and warm start

Note: the reference [CPnn] is linked to the PLCopen Coding Guidelines document of April 2016.

4.4 Reduction in Functions and Function Blocks

Standard Functions: (See IEC 61131-3; Tables 22 - 39). ST is only permitted in Extended Level

Description	**Basic Level**	**Extended Level**	**Comments**
AND	X	X	*Basic Level:* Operation of both BOOL and SAFEBOOL permitted at both levels. Three types of functions are designated to be used: 1) Only SAFEBOOL inputs and one SAFEBOOL output , 2) Only BOOL inputs and one BOOL output, 3) a mix of both for enabling functions: at least one SAFEBOOL input with at least one BOOL input and one SAFEBOOL output. *Extended Level:* Operation on ANY_BIT type is allowed. The function types 1) and 2) at *Basic Level* are allowed.

OR	X	X	*Basic level*: Only SAFEBOOL inputs and one SAFEBOOL output, or all BOOL. *Extended level*: Operation of both ANY_BIT and ANY_SAFEBIT permitted. Can use both types of OR function: 1) Only ANY_SAFEBIT inputs and one ANY_SAFEBIT output, 2) In case of one or more ANY_BIT inputs, the output is ANY_BIT
XOR, NOT	-	X	Operation of both ANY_BIT and ANY_SAFEBIT permitted. Operation for XOR is only allowed with 2 inputs. The output is ANY_SAFEBIT if and only if all inputs are ANY_SAFEBIT, otherwise ANY_BIT.
ADD, MUL, SUB, DIV, MOD, EXPT **+, *, -, /, MOD, ****	-	X	Operation of both INT/ DINT/ REAL and SAFEINT/ SAFEDINT/ SAFEREAL permitted. The output is of SAFE type if and only if all inputs are of SAFE type.
NEG **-**	-	X	Negation in ST (and FBD). Operation of both INT/ DINT/ REAL and SAFEINT/ SAFEDINT/ SAFEREAL permitted. The output is of SAFE type if and only if all inputs are of SAFE type.
MOVE	-	-	
SHL, SHR, ROR, ROL	-	-	Shift functions are not required, as binary information shall not be concatenated to BYTE/WORD.
EQ, NE =, <>	-	X	Operation of all data types except REAL, SAFEREAL permitted. The output is SAFEBOOL if and only if all inputs are of SAFE-type, otherwise BOOL. Comparison between floating point variables must use only <, <=, >, >=.
GT, GE,LE, LT >, >=, <=, <	-	X	Operation of both INT/DINT/REAL and SAFEINT/ SAFEDINT/ SAFEREAL permitted. The output is SAFEBOOL if and only if all inputs are of SAFE-type, otherwise BOOL.
SEL, MAX, MIN, LIMIT, MUX	-	X	Operation of all elementary data types is permitted. The output is of SAFE-type if and only if all inputs are of SAFE-type.
Type conversion functions (Figure 11 in IEC 61131-3 and tables 22 - 27)	X	X	Implicit type conversions are only allowed from SAFE to non_SAFE with the same datatype. Explicit type conversions are only allowed with the typed conversions <A>_TO_<B> and <A>_TRUNC_<B> (Table 22 in IEC61131-3). *Basic level*: Only from SAFE to non_SAFE conversion permitted. *Extended level*: For all data types that are supported if there is a run-time check that during the conversion the relevant data fits the target data type (no information is lost). The output is of SAFE-type if and only if all the input is of SAFE-type. In fig. 11 in IEC 61131-3 only the cells with “e” or “i” are allowed with the restriction of ANY_REAL to ANY_BIT and vice versa.
String functions	-	-	No STRING available.
Time functions	-	X	Only ADD, SUB, DIV, MUL with TIME or SAFETIME operands. The output is SAFETIME if and only if all inputs are of SAFE-type, otherwise TIME.
Unary REAL functions	-	X	E.g., SIN, SQRT, LOG.
Validate (Table 39 in IEC)	-	-	Not needed since invalid REAL numbers are excluded (see 4.3 Reduction in Data Types and Declarations)

Standard Function Blocks: (See IEC 61131-3; Tables 43 - 46)

Description	**Basic Level**	**Extended Level**	**Comments**

TON, TOF, TP	X	X	
CTU, CTD, CTUD	X	X	
Bistable FB (SR, RS)	-	X	No semaphores ("SEMA") permitted.
Edge detection	-	X	

4.5 ST Specific Reductions

Description	**Basic Level**	**Extended Level**	**Comments**
(expression)	-	X	Parenthesization The expression shall not be too complex. Use parenthesis to explicitly ex-press operation precedence (CGL L15)
Identifier (argument list)	-	X	Function evaluation
A := B; CV := CV+1; C:= ABS(X);	-	X	Assignment

Function Block Instance (…)	-	X	Function block call and output usage, e.g.
			CMD_TMR(IN := %IX5, PT := T#300ms);
			A := CMD_TMR.Q ;

RETURN;	-	X	Return statement

IF …
THEN …
ELSIF …
THEN …
ELSE …
END_IF
	-	X	For every IF instruction in the code, an ELSE clause should be present. (CGL L17)
			No more than 2 nested IF's

CASE … OF
…
ELSE …
END_CASE
	-	X	Case statement
			The ELSE branch is mandatory.
			No nesting of CASE is allowed.
			The CASE clauses shall be short, i.e., call a function

FOR … TO … BY … DO … END_FOR	-	X	For loop Only ANY_INT, ANY_SAFEINT allowed as counter, loop range must be constant. The loop variables shall not be modified within the FOR loop (CGL L12) Loop variables shall not be used outside the FOR loop (CGL L13)
WHILE … DO END_WHILE	-	-	While loop not supported. Danger of endless loop, loop count not constant
REPEAT … UNTIL … END_REPEAT	-	-	Repeat loop not supported. Danger of endless loop, loop count not constant
EXIT	-	X	Exit a loop. Usage of Continue and Exit instruction should be avoided (CGL L10)
CONTINUE	-	X	Do next Iteration. Usage of Continue and Exit instruction should be avoided (CGL L10)

Note: reference to CGL is a reference to the PLCopen Coding Guidelines of April 2016.

4.6 Other Reductions

(See IEC 61131-3; Tables 40, 73)

Description	**Basic Level**	**Extended Level**	**Comments**
Definition of FB	X	X	Basic Level: User Derived FBs for modularization purposes are permitted but shall be encoded only with Basic Level subset.
Directly represented variables	-	-	-
LD	X	X	See 4.2 Reduction in the Set of Programming Languages with the following restrictions for Basic Level: only power rails, 'normally open' contacts, and (normal) non-negated momentary coils are permitted.

FBD	X	X	See 4.2 Reduction in the Set of Programming Languages with the following restrictions for Basic Level: no negated inputs or outputs are permitted. In Function Block Diagram networks, one should avoid the assignment of variables between blocks [See CGL L2].
ST	-	X	See 4.2 Reduction in the Set of Programming Languages and 4.5 ST Specific Reductions with the restriction that it only can be used in Extended Level
SFC, IL	-	-	Only permitted on system level. Conforming to IEC 62061.
Other: C, C++, etc.	-	-	Only permitted on system level.
Multiple call of same PLCopen FB instance	-	-	The PLCopen Safety FBs must be processed once, and only once, every cycle. The same is valid for User Derived FBs which contain a PLCopen Safety FB or an internal state machine.

Multiple call of same IEC FB instance	-	X	The PLCopen Safety FBs must be processed once, and only once, every cycle. However, some standard IEC FBs (like TON) can be called several times in one cycle. Using this (only in Extended Level allowed) can increase the burden at the application program development or its certification.
			Example:
			fbTON();
			(* Timeout ? *)
			IF fbTON.Q THEN
			(* Restart Timeout with 100ms *)
			fbTON(IN:=FALSE);
			fbTON(IN:=TRUE; PT:= t#100ms);
			…
			END_IF

Implicit feedback loop in same network	-	X	The processing order of the FBs must be unique and transparent.
Explicit feedback loop in same network	-	X	The processing order of the FBs must be unique and transparent.
Multiple or conditional return	-	X	POUs shall have a single point of exit [CP14] Use only if it enhances the readability of the program.
Jumps, conditional jumps	-	X	To implement the state diagram. The use of Jumps should be avoided. Jumps must not result in unreachable code in the POU [CP2]
FB declaration features	-	-	See Table 40 of IEC 61131-3.
Program in Task	X	X	Cf. IEC 61131-3 table 63
Function block in Task	-	-	Tasks shall only call program POUs and not Function Blocks [CP16]
Programs without task association	-	-	Not permitted due to unreachable programs [CP2]

5 General Rules for Safety-Related Function Blocks

5.1 Function Block-Specific Rules

The following rules are applicable to all supplied safety related function blocks at library level.

Default signal	All safety-related Boolean I/O signals have the default safe condition "FALSE".
Signal level	The value of the SAFEBOOL is only applicable as follows: = 0 corresponds to safety as defined at system outputs. = 1 means that the safety aspects of the system are operating correctly, e.g., normal operation is possible. This representation reflects the functionality of the IEC 61131 environments, such as all outputs switch to "0" in the event of an error, as well as default value rules.
Missing input/output parameters	Missing parameters are permitted. Default values apply. These default values shall under no circumstances lead to an unsafe state. Default values are specified in the relevant FB specifications as *Initial Value*, including their attributes (VARIABLE or CONSTANT).
Start behavior	Initially the outputs are set to the default values. After the first call of the function blocks, the outputs are valid. There is a consistent start behavior, so there is no difference in the behavior between cold, warm, and hot start.
Timing diagrams	Timing diagrams, as shown at the FBs, are provided for explanation only. They do not represent the exact timing behavior. The exact timing behavior depends on the implementation (IF versus CASE).
Error handling and diagnostics	All safety-related function blocks have two error-related outputs: Error and DiagCode. These are provided for diagnostic purposes on the user application level, and not for diagnostics on the system/hardware level. The rule for safety-related environments is that the switching of a safety-related function has the highest priority, and following switching there is sufficient time for the diagnostics, either in the functional program or the operator interface.
FB Naming Conventions – Use of Prefix	All PLCopen-specified FBs are identified by the prefix **SF_**, and in PLCopen-compliant systems, the prefix **SF_** in POU-names is reserved for these FBs and their “derivates”. POU-name prefixes of the form **SF*x*_** are reserved to identify manufacturer-defined FBs, which conform to the common requirements of chapter 1-5 of the PLCopen specification.
FB Naming conventions - Extensions	Names of PLCopen-defined FBs can be extended by a suffix of the form **_*x***, e.g., SF_ESTOP_ABC. This is permissible and recommended for system-specific implementations and extensions of the diagnostic interface of the corresponding PLCopen-FB.
Naming conventions FBs with additional inputs	For system-specific diagnostic purposes it may be useful to extend a PLCopen-FBs by additional inputs and outputs. *If and only if* this new FB has the same PLCopen-specified behaviour on the original inputs and outputs independently from the use of the new inputs and outputs, it is *permissible* to give this FB the name of the original PLCopen-FB. However, it is *recommended* to name this FB by appending a suffix to the PLCopen-name, as described above.

Table 1: General rules

5.1.1 General Input Parameters

The following tables describe the name, type, and behavior of the generic FB interface:

Input Parameters		
Name	**Type**	**Description**
Activate	BOOL	Variable or constant. Activation of the FB. Initial value is FALSE. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the relevant safety device. This ensures no irrelevant diagnostic information is generated if a device is disabled. If FALSE, all output variables are set to the initial values. If no device is connected, a static TRUE signal must be assigned.
S_<safety-related input name>	SAFExxxx	Every SAFExxxx type input name begins with S_. Only variables may be assigned.
S_StartReset	SAFEBOOL	Variable or constant. FALSE (= initial value): Manual reset when PES is started (warm or cold). TRUE: Automatic reset when PES is started (warm or cold). This function shall only be activated if it is ensured that no hazard can occur at the start of the PES. Therefore, the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system

		or application measures to ensure that unexpected (or unintended) startup does not occur. It shall be noted in the FB manual that when using a SAFEBOOL variable additional validation of this application is necessary.
S_AutoReset	SAFEBOOL	Variable or constant. FALSE (= initial value): Manual reset when emergency stop button is released. TRUE: Automatic reset when emergency stop button is released. This function shall only be activated if it is ensured that no restart of the machine can occur through release of the emergency stop button. Therefore, the use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to ensure that unexpected (or unintended) restart of the machine does not occur. It shall be noted in the FB manual that when using a SAFEBOOL variable additional validation of this application is necessary.
Reset	BOOL	Variable. Initial value is FALSE. Depending on the function, this input can be used for different purposes: • Reset of the state machine and coupled error and status messages as indicated via DiagCode when the error cause has been removed. This reset behavior is designed as an acknowledge that the error is removed. • Manual reset of a "restart interlock" ("Wiederanlaufsperre" in German) by the operator (see EN 954-1). This reset behavior is designed as a functional reset. • Additional FB-specific reset functions. This function is only active on a signal change from FALSE to TRUE. A static TRUE signal causes no further actions but may be detected as an error in some FBs. The appropriate meaning must be described in every FB. It shall be noted in the FB manual that a SAFEBOOL must be connected instead of a BOOL depending on the safety requirements.

Table 2: Input parameters

5.1.2 General Output Parameters

Output Parameter		
Name	**Type**	**Description**
Ready	BOOL	If TRUE, indicates that the FB is activated, and the output results are valid (same as the "POWER" LED of a safety relay). If FALSE, the FB is not active, and the program is not executed. Useful in debug mode or to activate/deactivate additional FBs, as well as for further processing in the functional program.
S_<safety-related output name>	SAFExxxx	Every SAFExxxx data type output name begins with S_.
SafetyDemand	BOOL	Optional output indicating that the FB is active, and the primary safety function is demanded (e.g., related to the safety functionality). Other safety related input parameters are not considered (e.g., SafetyActive and EDM). The safety loop is not closed, and the safe state is demanded for the related safety output. There is no error. TRUE: Safety demand. FALSE: No Safety demand
ResetRequest	BOOL	Optional output which can be used to signal the operator to press the reset functionality to continue. TRUE: Reset requested FALSE: Reset not requested.
Error	BOOL	Error flag (same as "K1/K2" LED of a safety relay). When TRUE, indicates that an error has occurred, and the FB is in an error state. The relevant error state is mirrored at the DiagCode output. If FALSE, there is no error, and the FB is in another state. This again is mirrored by DiagCode (this means that DiagCode must be set in the same cycle as the state change).

		Useful in debug mode as well as for further processing in the functional program.
DiagCode	WORD	Diagnostic register. All states of the FB (Active, Not Active, and Error) are represented by this register. This information is encoded in hexadecimal format in order to represent more than 16 codes. Only one consistent code is represented at the same time. In the event of multiple errors, the DiagCode output indicates the first detected error. For additional information, see 5.2 Diagnostic Codes. Useful in debug mode as well as for further processing in the functional program.

Table 3: Output parameters

Note: Both SafetyDemand and ResetRequest set to TRUE do not provide unique information for the operator, and for this reason only one is SET at the same time. By providing these outputs directly in the FB, it is easy to connect these to an operator interface and in this way help to identify the applicable actions to be done. Both outputs are optional.

5.2 Diagnostic Codes

A transparent and unique diagnostic concept forms the basis of all function blocks. Thus, it is ensured that, regardless of the supplier's implementation, uniform diagnostic information is available to the user in the form of DiagCode. If no error is present, the internal status of the function block (state machine) is indicated. An error is indicated via a binary output (error). Detailed information about internal or external function block errors can be obtained via DiagCode. The function block must be reset via the different reset inputs.

Suppliers may add additional interfaces via function blocks with supplier-specific diagnostic information.

For all function blocks the following DIAG codes will be used to make the evaluation in software easier and more straightforward coupled to the outputs SafetyDemand and ResetRequest:

Name	DIAG	DiagCode$_{bin}$									Error	Safety Demand	Reset Request	Reset Error	Safety Outputs
		Nibble1			Nibble2			Nibble3	Nibble4						
		1	E	00	S	R	xx	xxxx	xxx	RE					
Error	Cyn0	1	1	00	0	0	xx	xxxx	000	0	1	0	0	0	0
Reset Error	Cyn1	1	1	00	0	0	xx	xxxx	000	1	1	0	0	1	0
Error AND ResetRequest	Cwn0	1	1	00	0	1	xx	xxxx	000	0	1	0	1	0	0
Error AND SafetyDemand	Not applicable (Error)														
		Nibble1			Nibble2			Nibble3	Nibble4						
		1	E	00	S	R	xx	xxxx	xxxx						
SafetyActive AND SafetyOutput	8yn0	1	0	00	0	0	xx	xxxx	0000		0	0	0	0	1
SafetyActive	8ynz	1	0	00	0	0	xx	xxxx	xxx0		0	0	0	0	0
Init AND ResetRequest	84n1	1	0	00	0	1	00	xxxx	0001		0	0	1	0	0
Init AND SafetyDemand	88n1	1	0	00	1	0	00	xxxx	0001		0	1	0	0	0
ResetRequest	84nz	1	0	00	0	1	00	xxxx	xxx0		0	0	1	0	0
SafetyDemand	88nz	1	0	00	1	0	00	xxxx	xxx0		0	1	0	0	0
															0
Idle	0000	0	0	00	0	0	00	0000	0000		0	0	0	0	0

Notes:

- S = 0 when only a reset is required. =1 when the safety link is not yet closed and needs operator attention. Equals the negation of the Safety Inputs.
- R = 0 when no reset is required. =1 when only a reset is required.
- RE = Reset Error
- x [0,1]
- n [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F] (In the combination 'yn', 'n' is leading over 'y', meaning that first 'n' is increased by one and after reaching 'F', 'y' is increased by one. Similar for the other combinations)
- y [0, 1, 2, 3]. However for error conditions in the muting FBs these are extended to [0..F] due to the many possible error conditions. In this case Nibble1 is set to C and Nibble4 is set to 4.
- z [2, 4, 6, 8, A, C, E]
- w [4, 5, 6, 7]

Generic Diagnostic Codes	
DiagCode	**Description**

0000_0000_0000_0000$_{\text{bin}}$
0000$_{\text{hex}}$
	The FB is not activated. This code represents the Idle state.
	For a generic example, the I/O setting for could be:
	Activate = FALSE
	S_In = FALSE or TRUE
	Ready = FALSE
	Error = FALSE
	S_Out = FALSE
	SafetyDemand = FALSE
	ResetRequest = FALSE

1000_0000_0000_0000$_{\text{bin}}$
8000$_{\text{hex}}$
	The FB is activated without an error or any other condition that sets the safety output to FALSE. This is the default operational state where the S_Out safety output = TRUE in normal operation. For a generic example, the I/O setting for could be:
	Activate = TRUE
	S_In = TRUE
	Ready = TRUE
	Error = FALSE
	S_Out = TRUE
	SafetyDemand = FALSE
	ResetRequest = FALSE

1000_0100_0000_0001$_{\text{bin}}$
8401$_{\text{hex}}$
	An activation has been detected by the FB and the FB is now activated, but the S_Out safety output is set to FALSE. This code represents the Init state of the operational mode. For a generic example, the I/O setting for could be:
	Activate = TRUE
	S_In = TRUE
	Ready = TRUE
	Error = FALSE
	S_Out = FALSE
	SafetyDemand = FALSE
	ResetRequest = TRUE

1000_0100_0000_0001$_{\text{bin}}$
8801$_{\text{hex}}$
	An activation has been detected by the FB and the FB is now activated, but the S_Out safety output is set to FALSE. This code represents the Init state of the operational mode. For a generic example, the I/O setting for could be:
	Activate = TRUE
	S_In = FALSE
	Ready = TRUE
	Error = FALSE
	S_Out = FALSE
	SafetyDemand = TRUE
	ResetRequest = FALSE

Generic Diagnostic Codes	
DiagCode	**Description**

1000_1000_0000_0010$_{\text{bin}}$
8802$_{\text{hex}}$
	The activated FB detects a safety demand ("Sicherheitsanforderung" in German), e.g., S_In = FALSE. The safety output is disabled. This is an operational state where the S_Out safety output = FALSE. For a generic example, the I/O setting for could be:
	Activate = TRUE
	S_In = FALSE
	Ready = TRUE
	Error = FALSE
	S_Out = FALSE
	SafetyDemand = TRUE
	ResetRequest = FALSE
	<Note: the detected safety demand refers to the states that are not IDLE or SAFESTATE>

1000_0100_0000_0011$_{\text{bin}}$
8403$_{\text{hex}}$
	The safety output of the activated FB has been disabled by a safety demand. The safety demand is now withdrawn, but the safety output remains FALSE until a reset condition is detected. This is an operational state where the S_Out safety output = FALSE. For a generic example, the I/O setting for could be:
	Activate = TRUE
	S_In = FALSE => TRUE (continuing with static TRUE)
	Ready = TRUE
	Error = FALSE
	S_Out = FALSE
	SafetyDemand = TRUE ==> FALSE
	ResetRequest = R

Table 4: General diagnostic code ranges

System or Device-Specific Codes	
DiagCode	**Description**
0xxx_xxxx_xxxx_xxxx$_{\text{bin}}$	X = System or device-specific message. This information contains the diagnostic information for the system or device and is mapped directly to the DiagCode output. (Note: 0000hex is reserved)

Table 5: System or device-specific codes

Notes: The Diagnostics Code 83FF is reserved for FBs which can be used in combination with the SF_ValveGroupControl as defined in PLCopen Safety Part 4 - Application Specific FBs for Presses, Version 1.0.
The Diagnostics Code 83FE is reserved for FBs which can be used in combination with the SF_TwoHandMultiOperator as defined in PLCopen Safety Part 4 - Application Specific FBs for Presses, Version 1.0.

5.3 Diagnostic FB

The function blocks provide detailed diagnosis information regarding errors and states and contain information about transition conditions that needs to be fulfilled by the operator before a state can be left. To determine if a Reset is necessary and or applicable the diagcode WORD needs to be evaluated by the standard control. For simpler implementations it would be helpful to have the information when a Reset is necessary, or a safety demand is required in general as binary information in the safety environment.

To support this, the outputs SafetyDemand and ResetRequest were proposed and accepted in 2012 in PLCopen Safety Part 3 – Extensions. Due to this there are 2 output interfaces as shown hereunder in Figure 6 and listed in 5.1.2 General Output Parameters. However, these outputs are optional and two different libraries can be delivered by the tool supplier: one with the support of these additional outputs and one without. There should be no name conflicts in programs using both options.

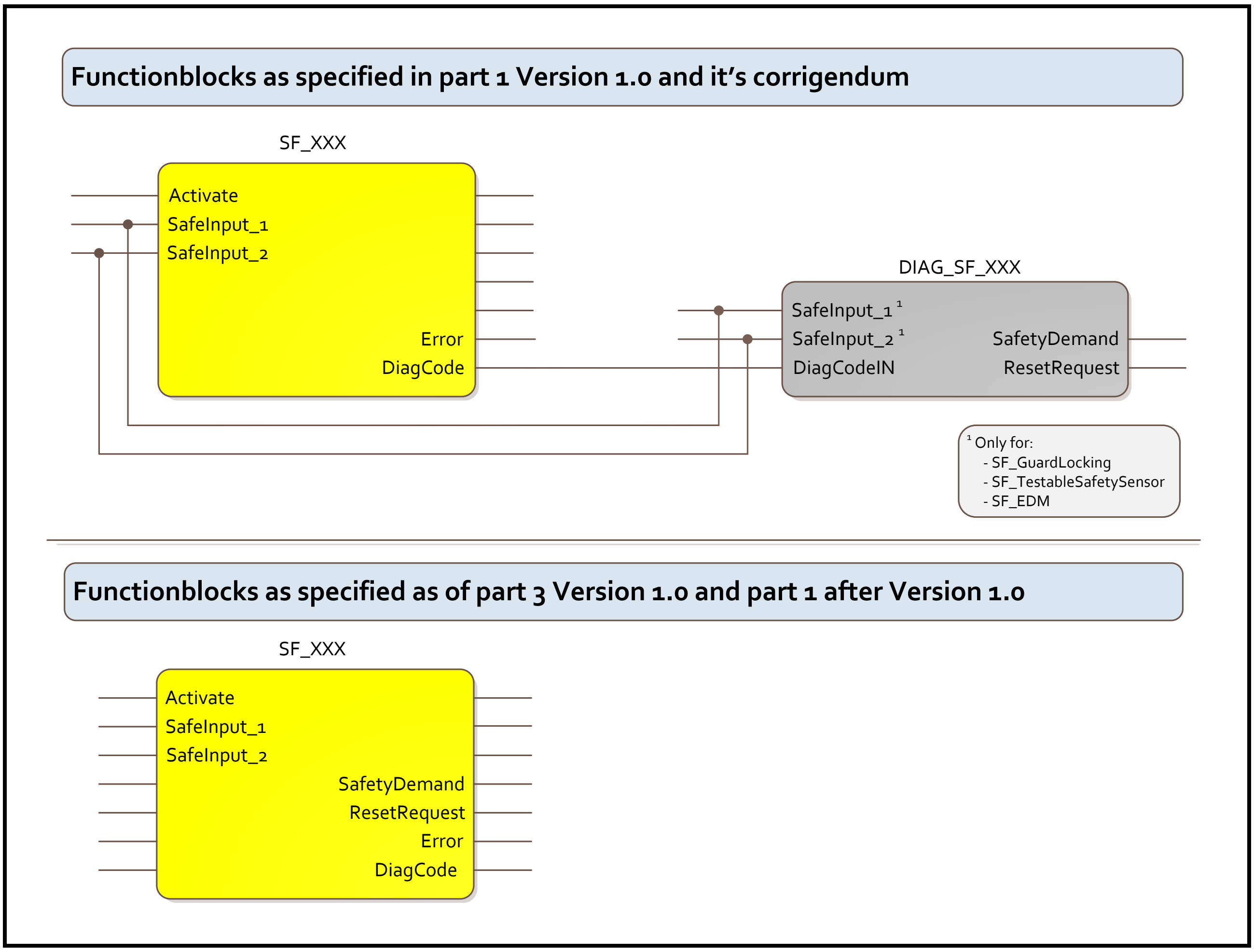


Figure 10: DIAG_SF_xxx and its function

5.4 Generic State Diagram

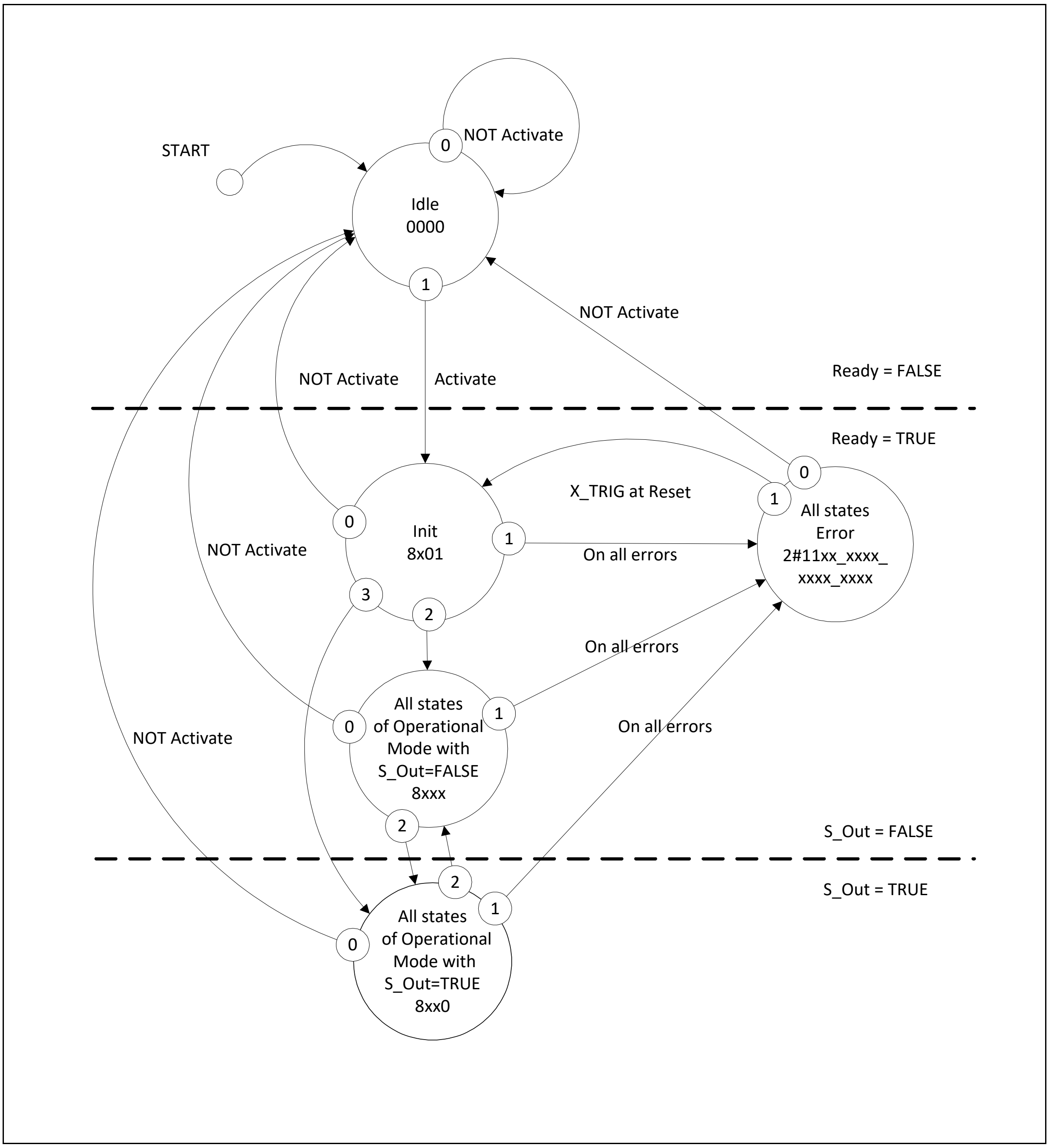


Figure 11: Generic state diagram of FBs

Explanation:

- The above diagram shows a general overview of the states and transitions. Some transitions are not named here, but have a meaning that is FB-specific, and are described with the relevant FBs.
- The diagram shows three areas: At the top the FB is not active and in the Safe state (safe outputs are FALSE), in the middle the FB is active and in the Safe state (safe outputs are FALSE), and at the bottom the FB is in the normal state, i.e., the safe outputs are TRUE.
- The first horizontal line in the state diagram shows the transition from a non-active FB to an active FB.
- The second horizontal line shows the transition from a non-safe state to a safe state of the FB.
- The priorities of possible parallel transitions are indicated by numbers (0 = highest priority).
- State bubbles contain the state name and hexadecimal DiagCode.
- Conditions OR, AND, XOR are used as logical operators and NOT is used as negation.

- The complete generic state diagram is omitted from the FB description. Within the FB description, the starting state is Idle, with the transitions to operational states via the Init state.
- The transition from any state due to Activate = FALSE, changes to Idle state (0 = highest priority reserved for Activate = FALSE) – for greater clarity, these transitions are not shown in each FB-related state diagram but are mentioned as a footnote to each state diagram.
- For reasons of clarity, the output setting is not described in the state diagram; an explicit truth table containing the "FB states to output(s)" information is part of each FB specification with the FB-specific error and status codes.
- Note to transition from 8xx0 to 0000: certain applications (like presses) need to finalize their cycle without the danger of any risk. In that case the transition can be delayed.

DiagCode	**State Name**	**State Description and Output Setting**
FB-specific error codes:		

Cxxx	Error	Ready = TRUE
		S_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

FB-specific status codes (no error):		

0000	Idle	Ready = FALSE
		S_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8x01	Init state of operational mode	Ready = TRUE
		S_Out = FALSE
		SafetyDemand = Depending
		ResetRequest = Depending
		Error = FALSE

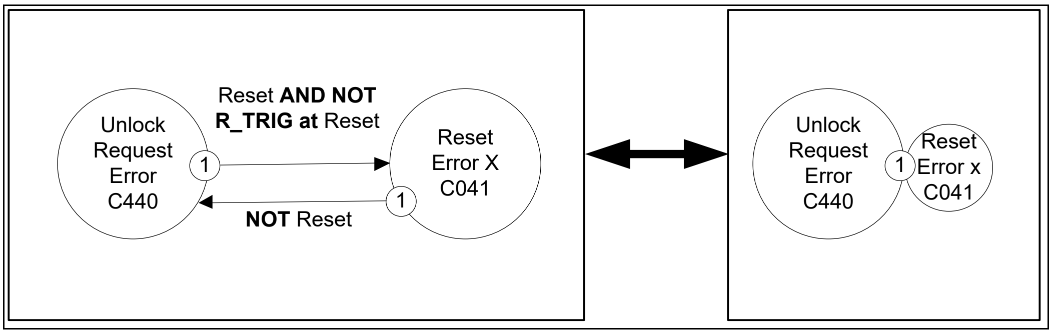
8xxx	All states of operational mode where S_Out = FALSE	Ready = TRUE
		S_Out = FALSE
		SafetyDemand = Depending
		ResetRequest = Depending
		Error = FALSE

8000	All states of operational mode where S_Out = TRUE	Ready = TRUE
		S_Out = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

Table 6: Function block codes of generic FBs

5.5 Simplified representation in the State Diagram

There are two simplifications used in the graphical representation. The additional outputs and the diagnostic codes reflect on the state diagram. To provide a clear overview, the following states are graphically merged in each state diagram: Reset Error.


Unlock Request Error C440
Reset AND NOT R_TRIG at Reset
1
Reset Error X C041
1
NOT Reset
Unlock Request Error C440
1
Reset Error x C041

The transition conditions are not shown, but always equal to the above. The priority is shown, and the DiagCode of the Reset Error. There is a relationship between the source of the transition (in this case Unlock Request Error) and the corresponding DiagCode: the second nibble is reused, e.g. C440 to C041.

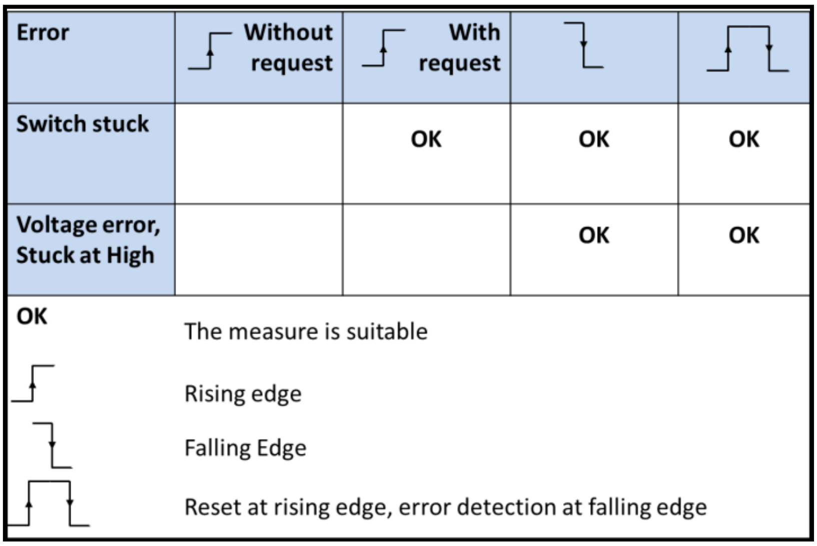
The second simplification deals with the merge of several different transition conditions in to one graphical arrow.
A state bubble can contain multiple DiagCodes. An arrow from a bubble with a single DiagCode in to such a bubble means a transition with a DiagCode as specified in the State Table. An arrow between two state bubbles with multiple DiagCodes means transitions from each state with DiagCode „n“ from bubble 1 to the corresponding DiagCode in bubble 2. The State Table gives more information on this.

5.6 Reset Behavior with ISO 13849-1:2015

Due to the acceptance of the ISO 13849-1 standard, the functionality of the manual reset is defined differently than within the PLCopen specifications: falling versus rising edge.
A falling edge manual reset signal (F_TRIG) is specified in ISO 13849-1 Ch. 5.2.2 while a rising edge manual reset signal (R_TRIG) is used in the PLCopen specification V1.0 (based on EN 954-1:1996).
Although further investigations by BG and TÜV have assessed these different functionalities as equivalent from the safety perspective, this opinion is not supported by all (new) assessors. This raises problems in the approval of installations in the field, which leads to on-going discussions on this matter or even disapproval of the installation.

In order to make the acceptance of an installation easier and faster, one of the main goals of the PLCopen Safety Specification, we need to add this behavior while staying backwards compatible for the existing implementations and installations. In this case we add a behavior where we check both the rising edge and the falling edge, called "Trailing Edge", as shown hereunder in addition to the other 2 possibilities:

- Rising Edge
- Falling Edge
- Trailing edge



Error	↑ Without request	↑ With request	↓	↑↓
Switch stuck		OK	OK	OK
Voltage error, Stuck at High			OK	OK

OK	The measure is suitable
↑	Rising edge
↓	Falling Edge
↑↓	Reset at rising edge, error detection at falling edge

Figure 12: Evaluation of reset switching behavior.

(Note: this schema is an unofficial translation of the German version in the document "Manuelle Rückstelleinrichtung" as provided by DGUV Fachbereich Holz und Metall in DGUV-Information 02/2015)
(Note: Trailing edge has minimum and maximum time)
(Note: the reset at rising edge and the error detection BOTH should be at the falling edge: no restart with an error)

5.6.1 Implementation and usage

This FB is specified separately but in normal operation always connected to a relevant FB which has a reset functionality. The separation makes it simpler for the implementer. However, towards the user it is better to encapsulate each pair of FBs and as such provide it as one entity. This means that also the Diagcodes need to be aligned (the Error outputs can be simply combined via an AND function). In the picture hereunder this alignment of DiagCodes is not shown in detail but just via the line on the bottom into the Safety FB. In practice a combination of AND and (exclusive) OR functionality will be sufficient to combine these.

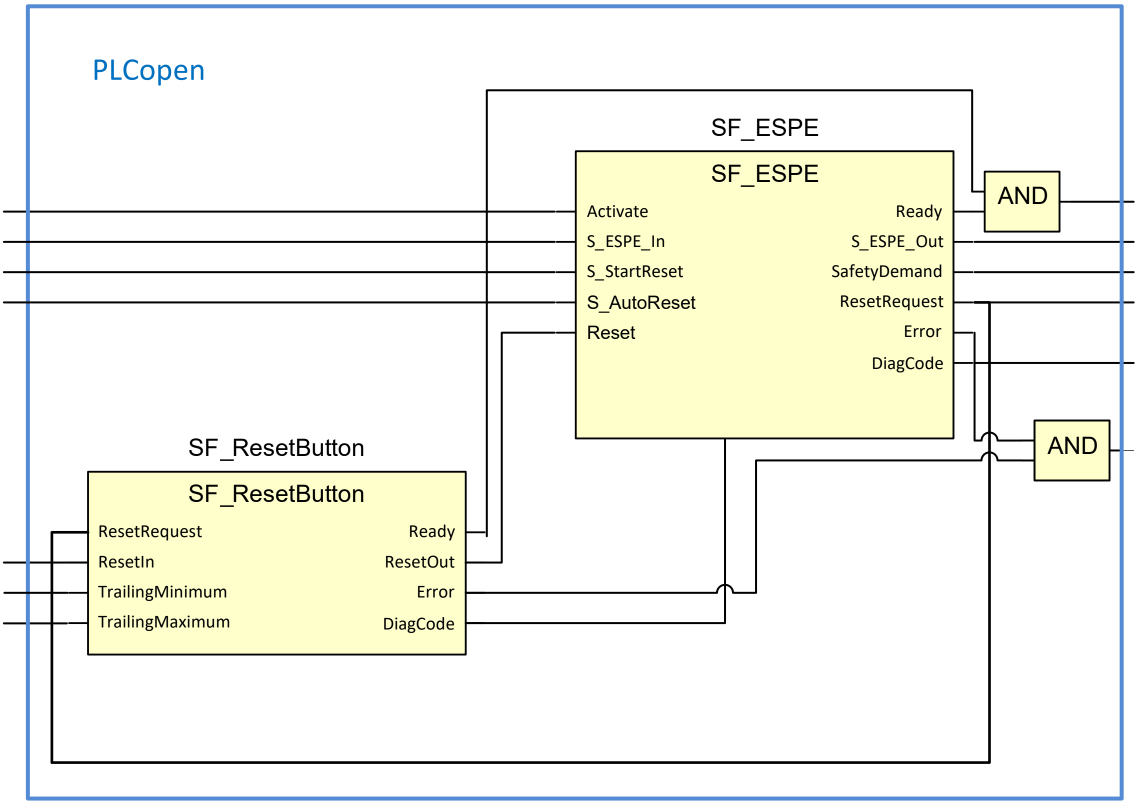


Figure 13: Example of embedded usage of SF_ResetButton

Due to the *ResetRequest* input the *SF_xxxxx* (*SF_ESPE* in the Example) can activate the *SF_ResetButton* Function Block. After this activation, the *SF_ResetButton* can detect a trailing edge. After the Detection the *SF_ResetButton* will issue a short Pulse on the *ResetOut* output which can be evaluated by the *SF_xxxxx* (*SF_ESPE* in the Example) Function Block as Reset.

5.6.2 SF_ResetButton always on

In this document the following situation should be discussed:

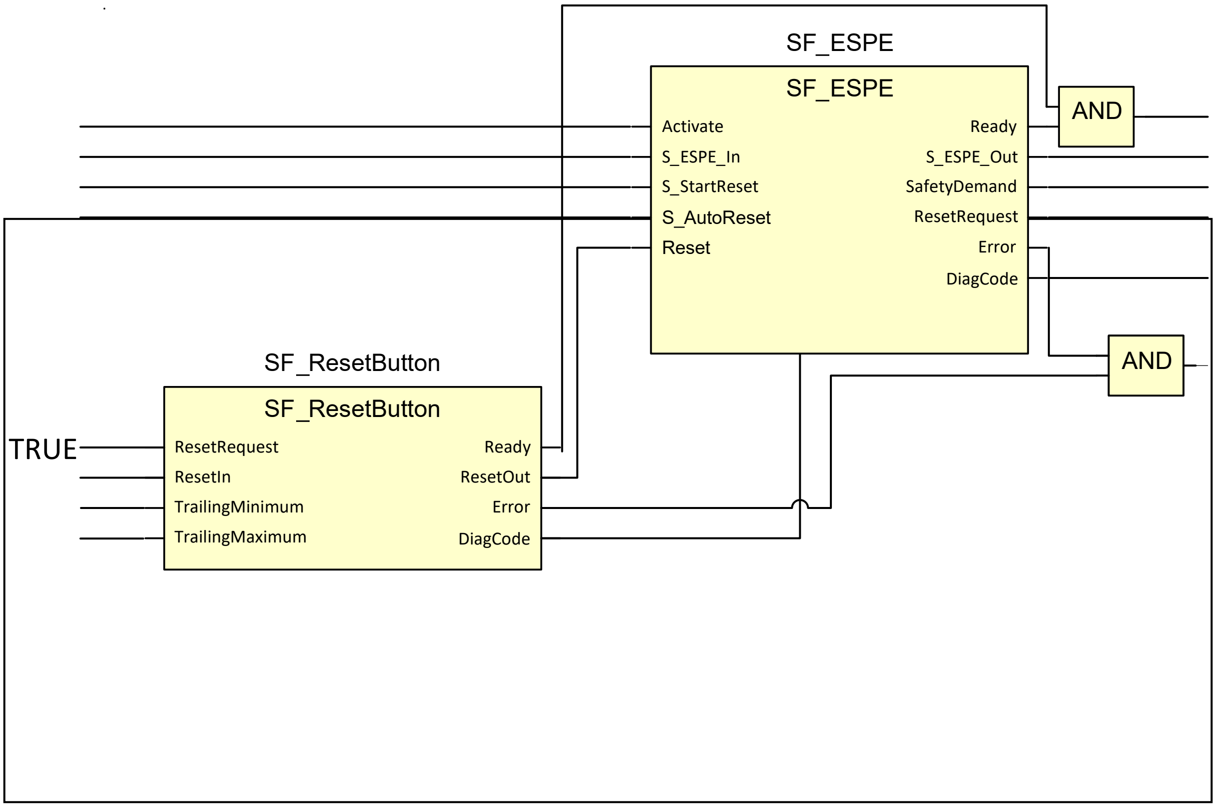


Figure 14: Example of separate usage of SF_ResetButton

In this case the *SF_ResetButton* will always evaluate the *ResetIn* Input and will issue the *ResetOut* Signal. The detection is independently of the internal state *SF_xxxxx* (*SF_ESPE* in the Example).

5.6.3 Compatibility to Part 1 Version 1.0

When one uses the function blocks as defined in Part 1 Version 1.0 only, one can use this Reset Button FB to switch to a negating edge triggering for the reset input.
In that case one must set the input ResetRequest continuously to TRUE.

6 Safety Function Blocks Pre-Processing

In this chapter the FBs are listed for the pre-processing phase conforming to Figure 6: Layers in the architectural model.

6.1 Reset Button

6.1.1 Applicable Safety Standards

Standards	Requirements
EN ISO 13849-1:2015	5.2.2 Manual reset function (See also 5.6 Reset Behavior with ISO 13849-1:2015)

Additional information:
The requirements of the manual reset functions are:

(a) be provided through a separate and manually operated device within the SRP/CS,	✓
(b) only be achieved if all safety functions and safeguards are operative,	FB
(c) not initiate motion or a hazardous situation by itself,	✓
(d) be by deliberate action,	FB
(e) enable the control system for accepting a separate start command,	✓
(f) only be accepted by disengaging the actuator from its energized (on) position.	FB

Only the requirements marked as “FB” can be influenced by the Timing / Configuration of the *SF_ResetButton*. Only these topics will be discussed in further Details.

It is assumed that the Function Block which is assigned to the Safety Function / Safety Guard only reacts on the *ResetEvaluation* Output if the Safety Function / Safety Guard is in operation. If the Safety Function / Safety Guard is not in Operation, the assigned Function Block will ignore the *ResetEvaluation* Output.

The Intention of the Standard is the stated deliberate Action (d). Even only a “disengaging of the actuator” (falling edge) is mentioned, the intention of the standard is a LOW – HIGH – LOW Signal by the actuator (deliberate action).

Taking the intention in consideration the *SF_ResetButton* Function Block realizes only a Trailing Edge Functionality. So, a LOW – HIGH – LOW Signal is always required.

The *SF_ResetButton* Function Block generates the *ResetEvaluation* Signal at the falling edge.

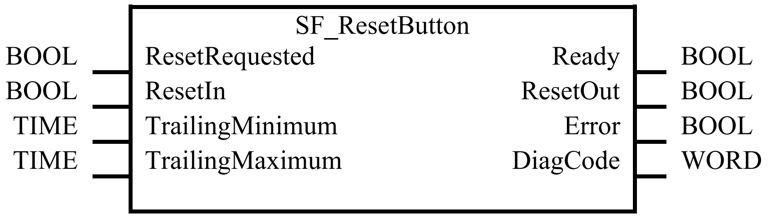
If, e.g., by locking the actuator for the reset, the *TrailingMaximum* Time is exceeded, the *SF_ResetButton* Function Block will generate an error. If the Signal of the actuator is too short, the *ResetEvaluation* Output will always be LOW.

In Chapter 5.6 Reset Behavior with ISO 13849-1:2015 Figure 12: Evaluation of reset switching behavior.**Error! Reference source not found.** is an example for the “worst case” timing. Here the requirements (b) and (f) of the ISO 13849-1 are always fulfilled.

6.1.2 Interface Description

FB Name	**SF_ResetButton**		
This function block adds the trailing edge functionality to all the function blocks with reset input with rising edge detection. This can be used to comply to EN ISO 13849-1:2015			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
ResetRequested	BOOL	TRUE	Input which should be connected to the ResetRequest output of the paired FB. TRUE: ResetRequested FALSE: No reset requested / no monitoring of ResetIn.
ResetIn	BOOL	FALSE	Variable. Input of reset button. FALSE: reset button released. TRUE: reset button actuated by operator.

TrailingMinimum	TIME	T#350ms	Constant. Valid in trailing mode. Minimum time that the reset switch must be actuated. If the reset button is pushed shorter than this time, the reset is ignored. Typical value 350msec.Absolut minimum value is 100msec.Minimum value 2 PLC cycles.
TrailingMaximum	TIME	T#2s	Constant. Valid in trailing mode. Maximum time that the reset switch is actuated. Typical value can be around 2 sec. If the reset button is pushed longer than this time, the reset is ignored.
VAR_OUTPUT			
Ready	BOOL	FALSE	See 5.1.2 General Output Parameters
ResetOut	BOOL	FALSE	Pulse for the initiation of the reset procedure. This pulse is generated after the falling edge. Pulse output with rising edge first. At least 1 cycle.
Error	BOOL	FALSE	See 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See 5.1.2 General Output Parameters
Notes: 1. The output ResetEvaluation is a rising edge (R_TRIG) pulse of length of (at least?) 1 cycle, which can be connected to any FB which has a reset input, and as such is compliant to the PLCopen Safety FBs version 1.0. 2. The ResetRequest input is connected to the ResetRequest output of the relevant FB. With this the timing interval is controlled during which the Reset Button needs to be checked. 3. This FB uses AutoReset			



6.1.3 Functional description

For the functional description refer to 5.6 Reset Behavior with ISO 13849-1:2015.

State Diagram

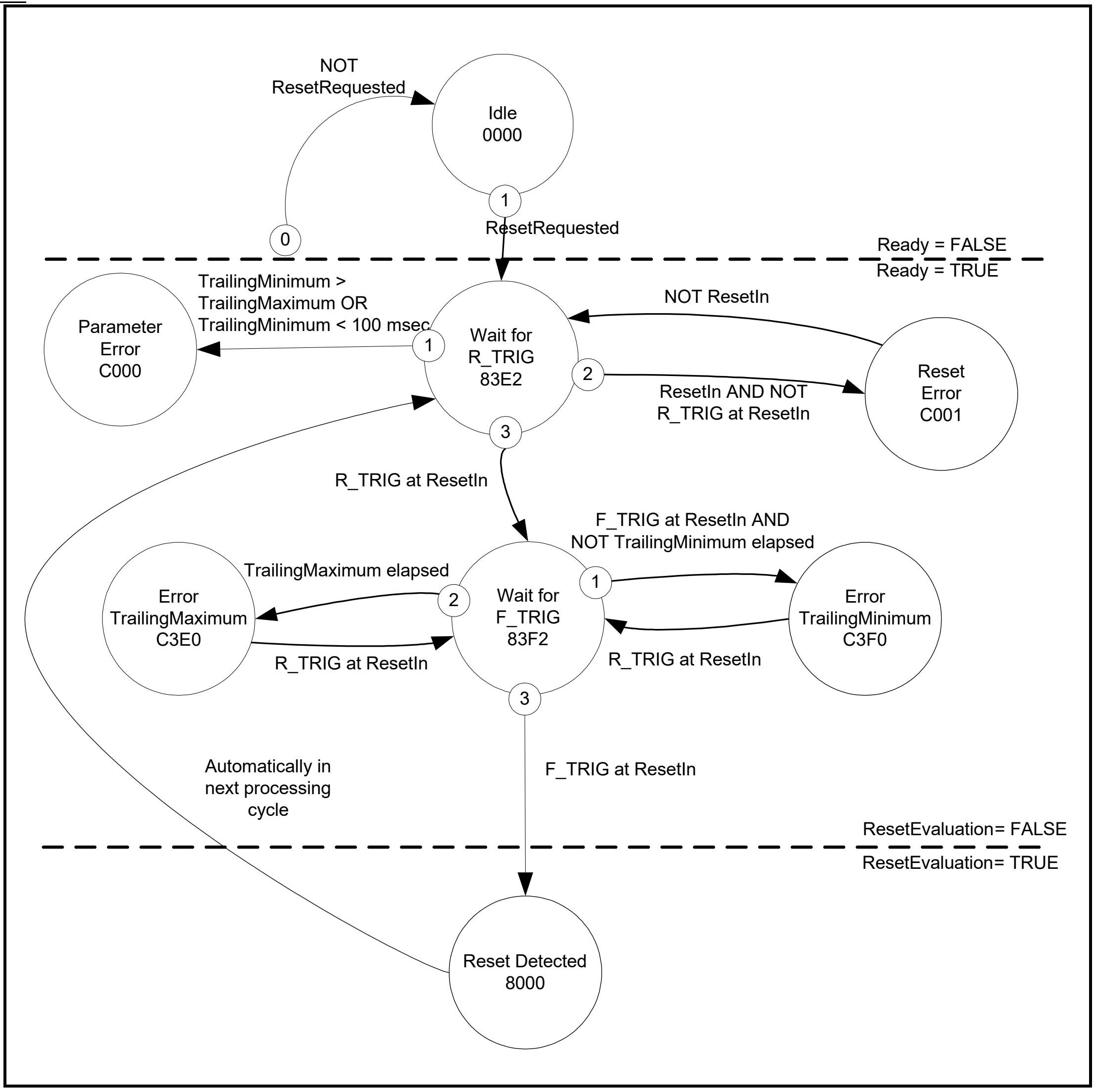


Figure 15: State diagram for SF_ResetButton

Typical Timing Diagram

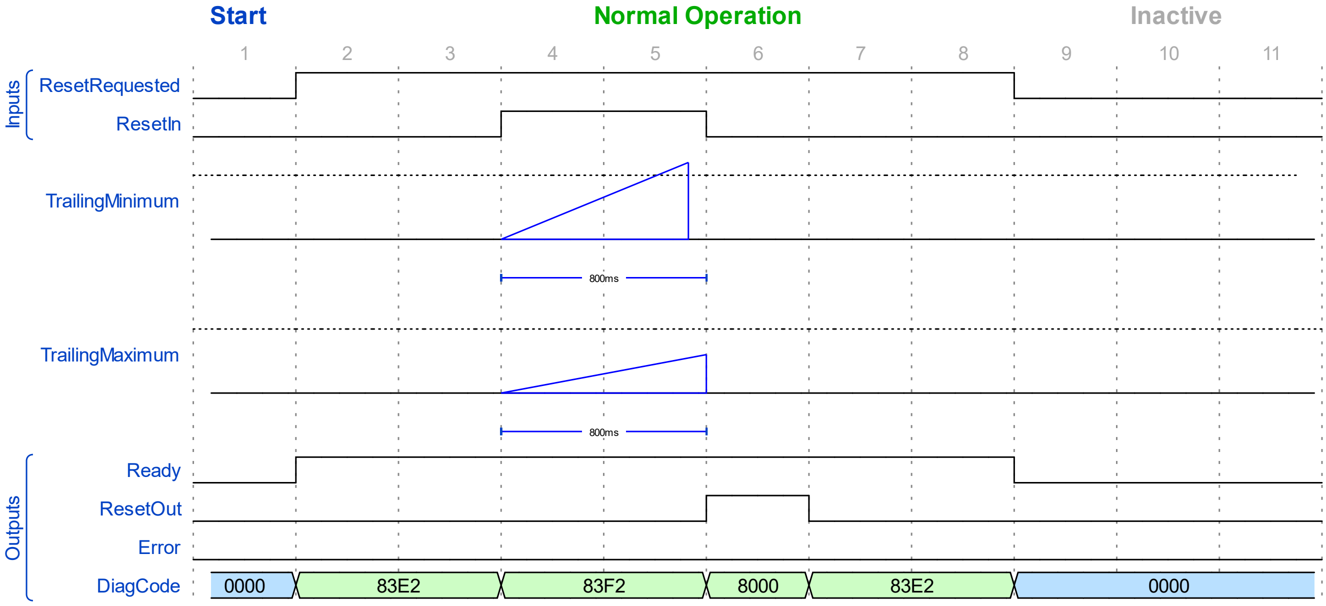


Figure 16: Timing example of Trailing Edge

Notes: Chapter 5.2.2 of the ISO 13849-1 mentioned only the falling edge. So, if the Falling Edge comes at the same time as the Safety Function / Safety Guard is operative the manual reset can be accepted.
(*) The Safety Function / Safety Guard Operative is the internal status of the FB according to the ISO 13849-1:2015, Chapter 5.2.2 Manual Reset Function second bullet point (only be achieved if all safety functions and safeguards are operative). In this example this is represented by the S_ESPE_Out.

6.1.4 Error Detection

If the ResetIn is TRUE when ResetRequested becomes TRUE, an Error is generated.
If the input ResetRequested is TRUE and the ResetIn is TRUE and the time input TrailingMinimum is not reached or the input TrailingMaximum is exceeded an error is detected.

6.1.5 Error Behavior

In case of a static TRUE signal at the ResetIn input, the DiagCode output indicates the relevant error code, and the Error output is set to TRUE.

6.1.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C000	Parameter Error	TrailingMinimum > TrailingMaximum OR TrailingMinimum < 100 msec.
		Ready = TRUE
		ResetOut = FALSE
		Error = TRUE

C001	Reset Error	ResetIn is TRUE while waiting for NOT ResetIn.
		Ready = TRUE
		ResetOut = FALSE
		Error = TRUE

C3E0	Error Trailing Maximum	TrailingMaximum elapsed before detecting F_TRIG at ResetIn. Waiting for R_TRIG at ResetIn.
		Ready = TRUE
		ResetOut = FALSE
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

C3F0	Error Trailing Minimum	F_TRIG at ResetIn detected before TrailingMinimum elapsed. Waiting for R_TRIG at ResetIn.
		Ready = TRUE
		ResetOut = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		ResetOut = FALSE
		Error = FALSE

83E2	Wait for R_TRIG	The function block is enabled. Wait for R_TRIG at ResetIn.
		Ready = TRUE
		ResetOut = FALSE
		Error = FALSE

83F2	Wait for F_TRIG	ResetIn is TRUE. Wait for F_TRIG at ResetIn
		Ready = TRUE
		ResetOut = FALSE
		Error = FALSE

8000	Reset Detected	Valid reset behavior was detected.
		The state is valid for at least one cycle and will automatically transfer to 83E2.
		Ready = TRUE
		ResetOut = TRUE
		Error = FALSE

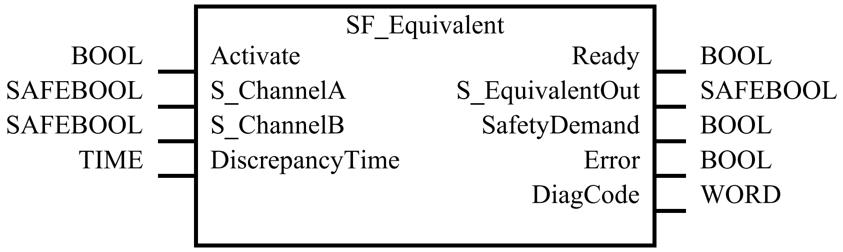
6.2 Equivalent

6.2.1 Applicable Safety Standards

Standards	Requirements
EN ISO 13849-1:2015	6.2.6 Category 3 6.2.7 Category 4 Appendix E.1

6.2.2 Interface Description

FB Name	**SF_Equivalent**		
This function block converts two equivalent SAFEBOOL inputs (both NO or NC) to one SAFEBOOL output, including discrepancy time monitoring. This FB should not be used stand-alone since it has no restart interlock. It is required to connect the output to other safety related functionalities.			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
S_ChannelA	SAFEBOOL	FALSE	Variable. Input A for logical connection. FALSE: Contact A open TRUE: Contact A closed.
S_ChannelB	SAFEBOOL	FALSE	Variable. Input B for logical connection. FALSE: Contact B open TRUE: Contact B closed.
DiscrepancyTime	TIME	T#0ms	Constant. Maximum monitoring time for discrepancy status of both inputs.
VAR_OUTPUT			
Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
S_EquivalentOut	SAFEBOOL	FALSE	Safety related output FALSE: Minimum of one input signal = "FALSE" or status change outside of monitoring time. TRUE: Both input signals "active" and status change within monitoring time.
SafetyDemand	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters
Notes: For certain (lower) levels of safety requirements it can be allowed to use BOOL as inputs and SAFEBOOL as output. However, this has to be evaluated via the FMEA of the application. In the library there should be made a distinction between the SAFEBOOL and BOOL version.			

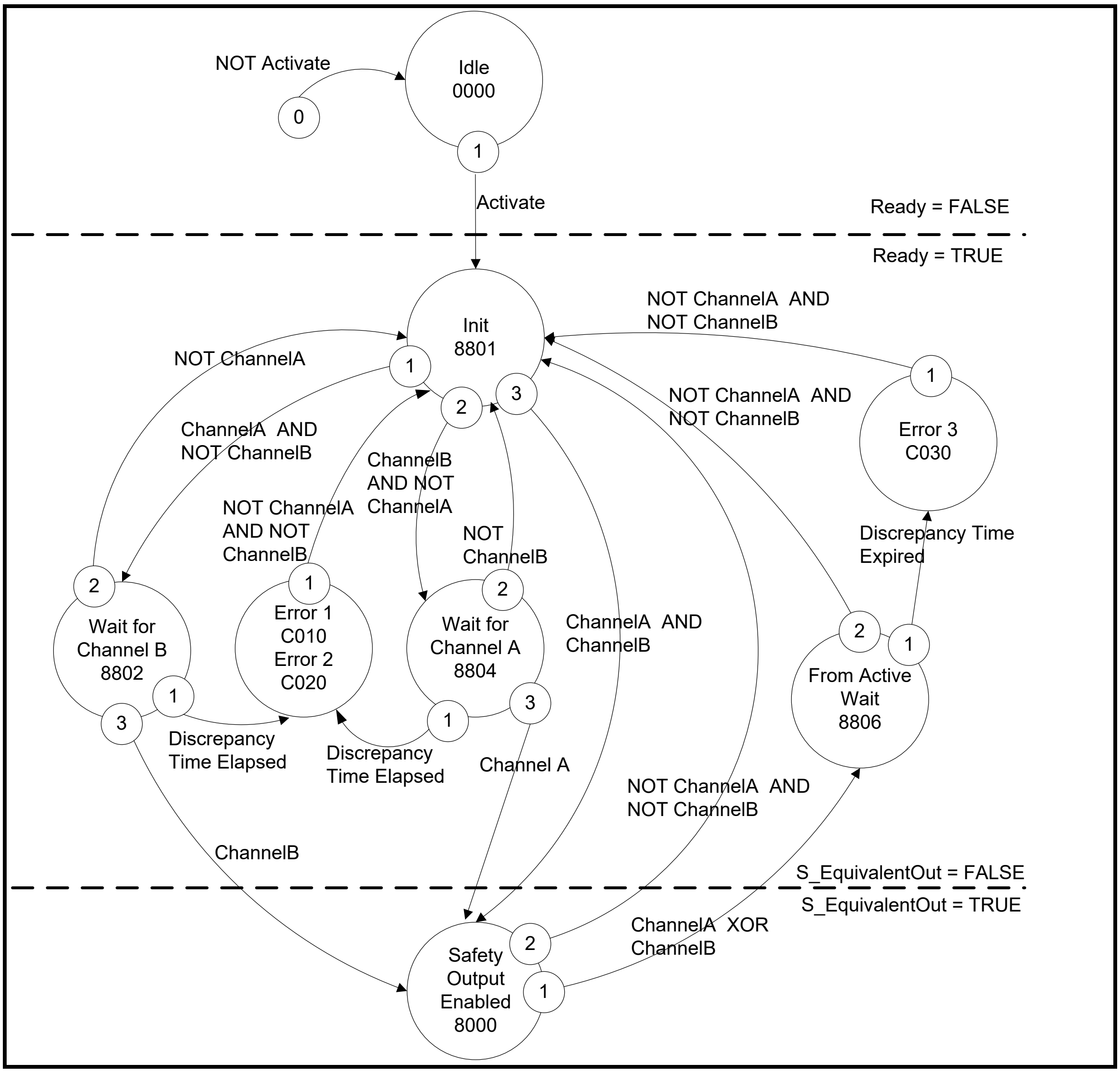


6.2.3 Functional Description

This function block converts two equivalent SAFEBOOL inputs to one SAFEBOOL output with discrepancy time monitoring. Both input Channels A and B are interdependent. The function block output shows the result of the evaluation of both channels.

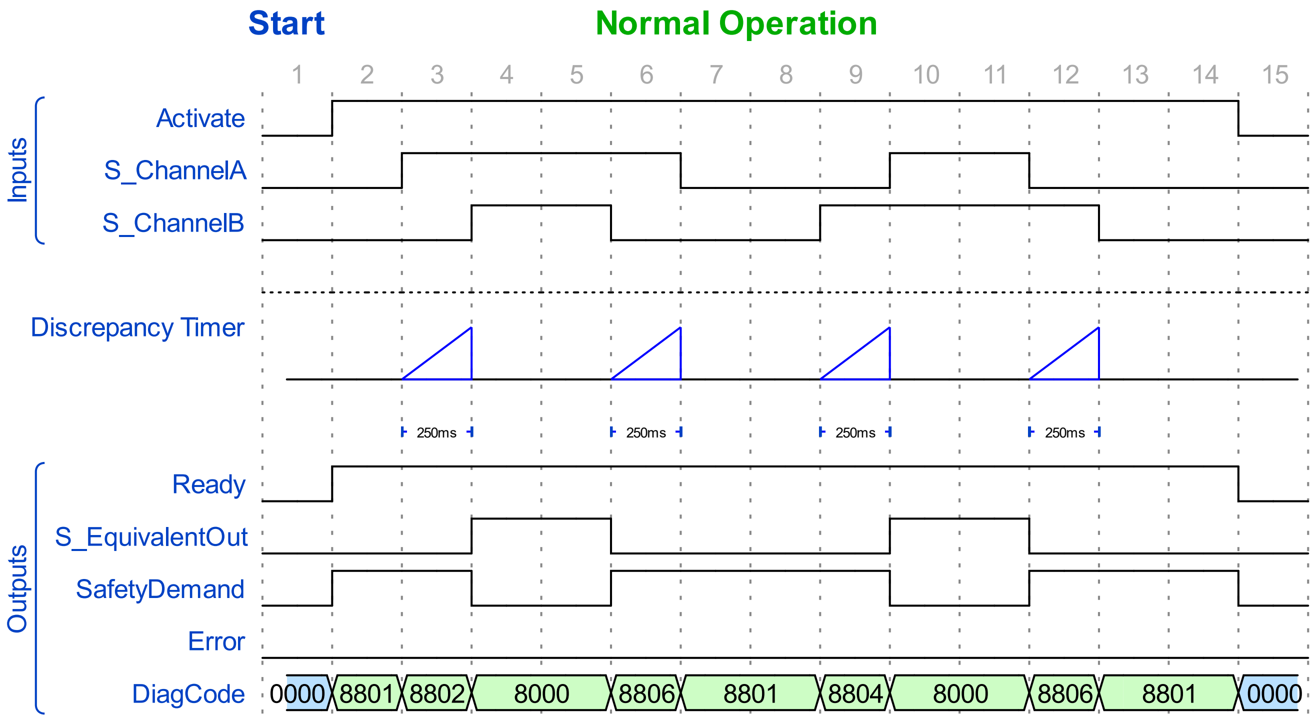
If one channel signal changes from TRUE to FALSE, the output immediately switches off (FALSE) for safety reasons.
Discrepancy time monitoring: The discrepancy time is the maximum period during which both inputs may have different states without the function block detecting an error. Discrepancy time monitoring starts when the status of an input changes. The function block detects an error when both inputs do not have the same status once the discrepancy time has elapsed.
The inputs must be switched symmetrically. This means that monitoring is performed for both the switching on process as well as the switching off process.

State Diagram



Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).
Figure 17: State diagram for SF_Equivalent

Typical Timing Diagrams



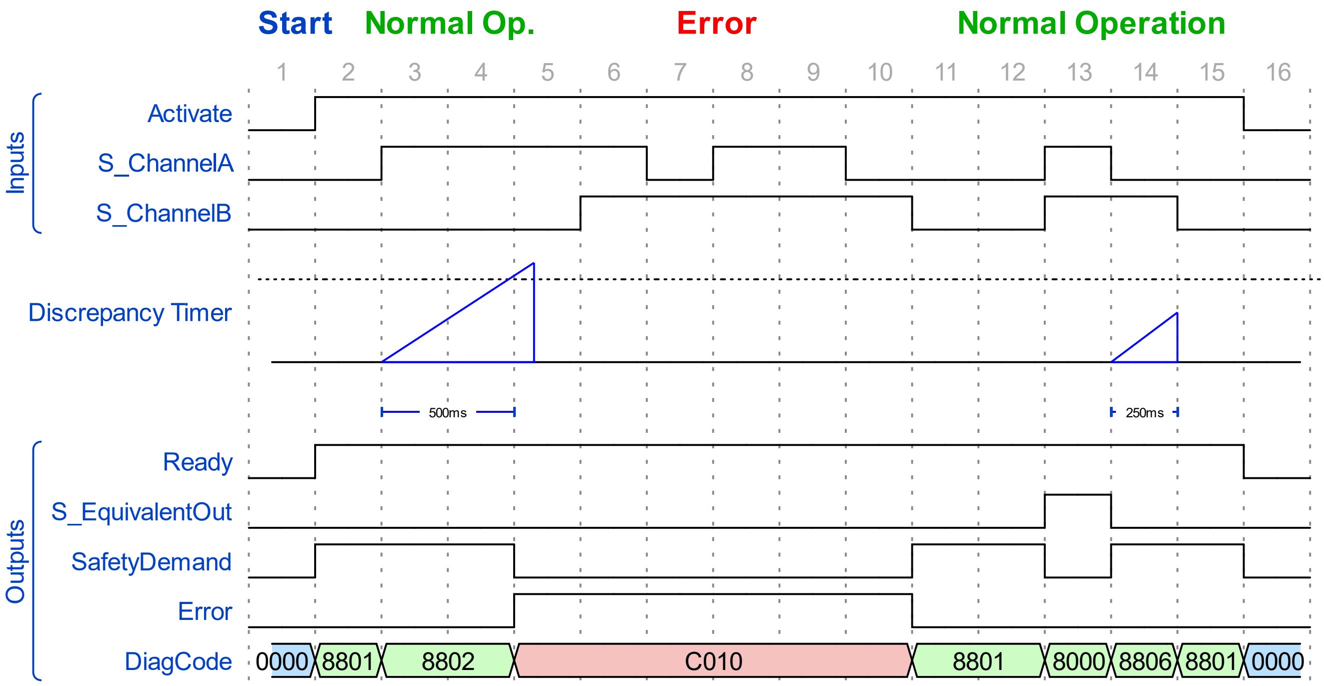


Figure 18: Timing diagrams for SF_Equivalent

6.2.4 Error Detection

The function block monitors the discrepancy time between Channel A and B, when switching to TRUE and also when switching to FALSE.

6.2.5 Error Behavior

S_EquivalentOut is set to FALSE. Error is set to TRUE. DiagCode indicates the Error states. There is no Reset defined as an input coupled with the reset of an error. If an error occurs in the inputs, a new set of inputs with correct S_EquivalentOut must be able to reset the error flag. (Example: if a switch is faulty and replaced, using the switch again results in a correct output)

6.2.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C010	Error 1	Discrepancy time elapsed in state 8802.
		Ready = TRUE
		S_EquivalentOut = FALSE
		SafetyDemand = FALSE
		Error = TRUE

C020	Error 2	Discrepancy time elapsed in state 8804.
		Ready = TRUE
		S_EquivalentOut = FALSE
		SafetyDemand = FALSE
		Error = TRUE

C030	Error 3	Discrepancy time elapsed in state 8806.
		Ready = TRUE
		S_EquivalentOut = FALSE
		SafetyDemand = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_EquivalentOut = FALSE
		SafetyDemand = FALSE
		Error = FALSE

8801	Init	An activation has been detected by the FB and the FB is now activated.
		Ready = TRUE
		S_EquivalentOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

8000	Safety Output Enabled	The inputs switched to TRUE in equivalent mode.
		Ready = TRUE
		S_EquivalentOut = TRUE
		SafetyDemand = FALSE
		Error = FALSE

8802	Wait for Channel B	Channel A has been switched to TRUE - waiting for Channel B; discrepancy timer started.
		Ready = TRUE
		S_EquivalentOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

8804	Wait for Channel A	Channel B has been switched to TRUE - waiting for Channel A; discrepancy timer started.
		Ready = TRUE
		S_EquivalentOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

8806	From Active Wait	One channel has been switched to FALSE; waiting for the second channel to be switched to FALSE, discrepancy timer started.
		Ready = TRUE
		S_EquivalentOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

6.3 Antivalent

6.3.1 Applicable Safety Standards

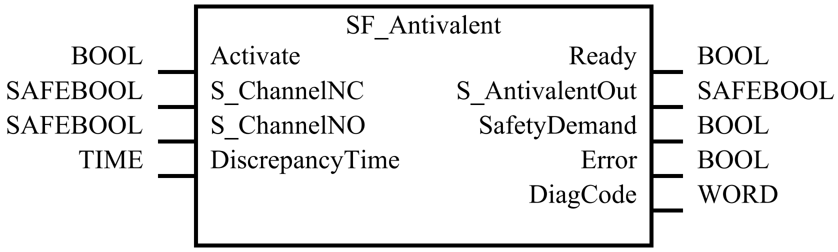
Standards	**Requirements**
EN ISO 13849-1:2015	6.2.6 Category 3 6.2.7 Category 4 Appendix E.1

6.3.2 Interface Description

FB Name	**SF_Antivalent**		
This function block converts two antivalent SAFEBOOL inputs (NO/NC pair) to one SAFEBOOL output with discrepancy time monitoring. This FB should not be used stand-alone since it has no restart interlock. It is required to connect the output to other safety related functionalities.			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
S_ChannelNC	SAFEBOOL	FALSE	Variable. NC stands for Normally Closed. Input for NC connection. FALSE: NC contact open. TRUE: NC contact closed.
S_ChannelNO	SAFEBOOL	TRUE	Variable. NO stands for Normally Open. Input for NO connection. FALSE: NO contact open TRUE: NO contact closed
DiscrepancyTime	TIME	T#0ms	Constant. Maximum monitoring time for discrepancy status of both inputs.
VAR_OUTPUT			
Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
S_AntivalentOut	SAFEBOOL	FALSE	Safety related output FALSE: Minimum of one input signal "not active" or status change outside of monitoring time. TRUE: Both inputs signals "active" and status change within monitoring time.
SafetyDemand	BOOL	FALSE	See 5.1.2 General Output Parameters
Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters

Notes:

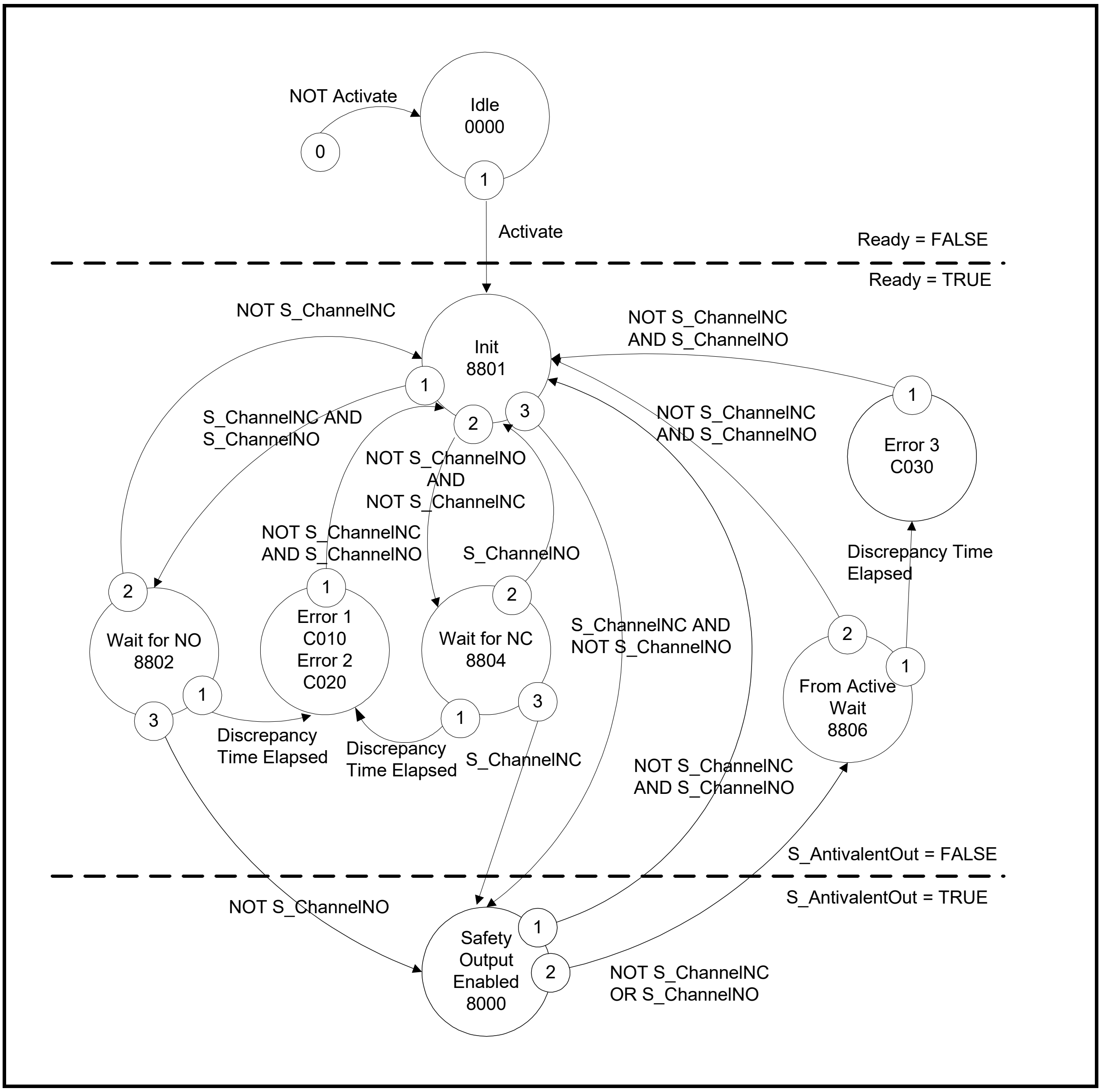
- "Antivalent" means that during normal operation, the two inputs are in opposite states at the same time. This is sometimes called "complementary" or "non-equivalent".
- For certain (lower) levels of safety requirements it can be allowed to use BOOL as inputs and SAFEBOOL as output. However, this has to be evaluated via the FMEA of the application. In the library there should be made a distinction between the SAFEBOOL and BOOL version.



6.3.3 Functional Description

This function block converts two antivalent SAFEBOOL inputs to one SAFEBOOL output with discrepancy time monitoring. Both input channels are interdependent. The function block output shows the result of the evaluation of both channels.
If S_AntivalentOut = TRUE and one of the safety related inputs changes, the output immediately switches to FALSE.
Discrepancy time monitoring: The discrepancy time is the maximum period during which both inputs may have the same states (i.e., both inputs are either TRUE or FALSE) without the function block detecting an error. Discrepancy time monitoring starts when the status of an input changes. The function block detects an error when both inputs do not have antivalent values once the discrepancy time has elapsed.
The inputs must be switched symmetrically. This means that monitoring is performed for both the switching on process as well as the switching off process.

State Diagram



Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Figure 19: State diagram for SF_Antivalent

Typical Timing Diagrams

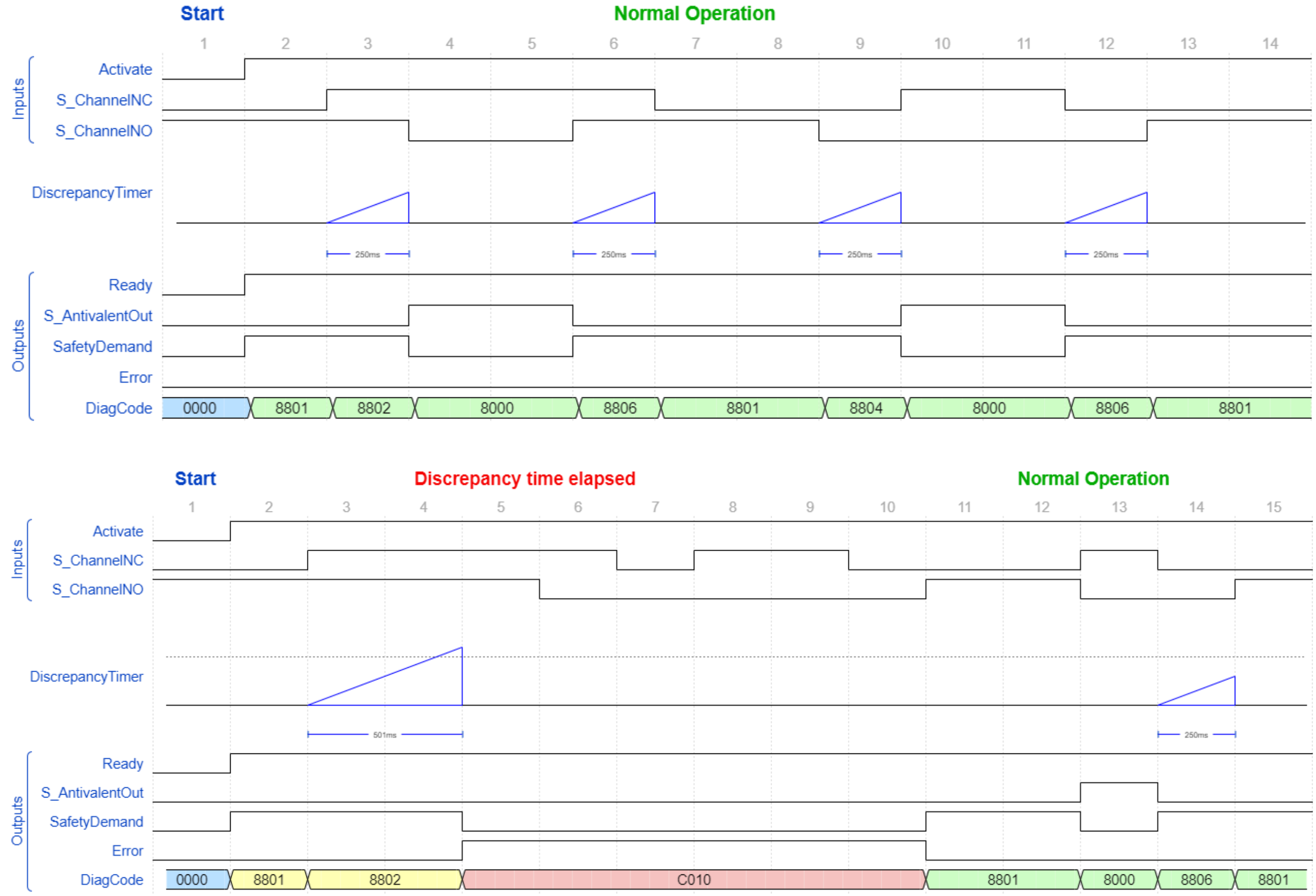


Figure 20: Timing diagrams for SF_Antivalent

6.3.4 Error Detection

The function block monitors the discrepancy time between Channel NO and Channel NC.

6.3.5 Error Behavior

The output S_AntivalentOut is set to FALSE. Error is set to TRUE. DiagCode indicates the Error states.

There is no Reset defined as an input coupled with the reset of an error. If an error occurs in the inputs, one new set of inputs with the correct value must be able to reset the error flag. (Example: if a switch is faulty and replaced, using the switch again results in a correct output)

6.3.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C010	Error 1	Discrepancy time elapsed in state 8802.
		Ready = TRUE
		S_AntivalentOut = FALSE
		SafetyDemand = FALSE
		Error = TRUE

C020	Error 2	Discrepancy time elapsed in state 8804.
		Ready = TRUE
		S_AntivalentOut = FALSE
		SafetyDemand = FALSE
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

C030	Error 3	Discrepancy time elapsed in state 8806.
		Ready = TRUE
		S_AntivalentOut = FALSE
		SafetyDemand = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_AntivalentOut = FALSE
		SafetyDemand = FALSE
		Error = FALSE

8801	Init	An activation has been detected by the FB and the FB is now activated.
		Ready = TRUE
		S_AntivalentOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

8000	Safety Output Enabled	The inputs switched to the Active state in antivalent mode.
		Ready = TRUE
		S_AntivalentOut = TRUE
		SafetyDemand = FALSE
		Error = FALSE

8802	Wait for NO	ChannelNC has been switched to TRUE - waiting for ChannelNO to be switched to FALSE; discrepancy timer started.
		Ready = TRUE
		S_AntivalentOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

8804	Wait for NC	ChannelNO has been switched to FALSE - waiting for ChannelNC to be switched to TRUE; discrepancy timer started.
		Ready = TRUE
		S_AntivalentOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

8806	From Active Wait	One channel has been switched to inactive; waiting for the second channel to be switched to inactive too.
		Ready = TRUE
		S_AntivalentOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

6.4 Mode Selector

6.4.1 Applicable Safety Standards

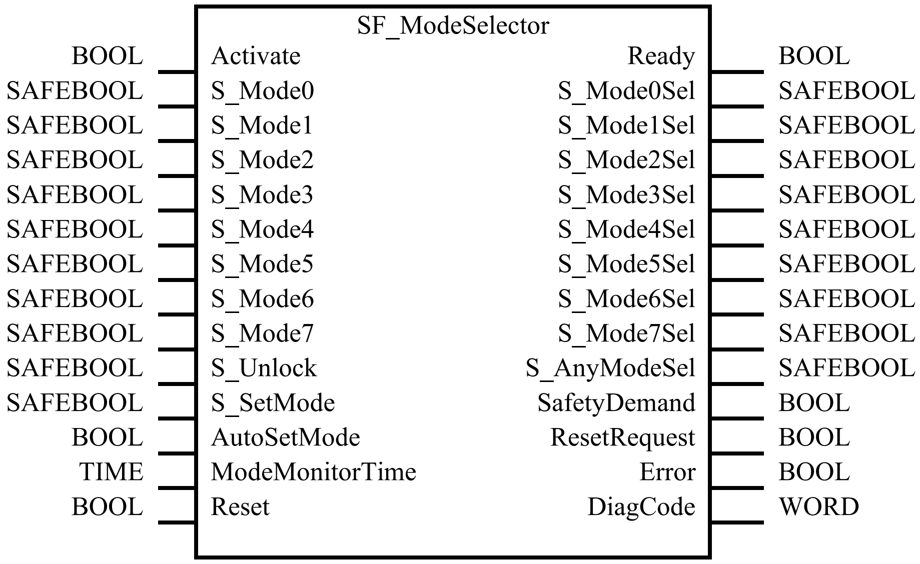
Standards	Requirements
MRL 2006/42/EC, Annex I	1.2.3. Starting ... It must be possible to start machinery only by voluntary actuation of a control provided for the purpose.... The same requirement applies:... - when effecting a significant change in the operating conditions.... 1.2.5 ... mode selector which can be locked in each position. Each position of the selector must correspond to a single operating or control mode....
EN ISO 12100:2010	6.2.11.4: Restart following power failure/spontaneous restart. 6.2.11.10 Selection of Control and Operating Modes … shall be fitted with a mode selector which can be locked in each position. Each position of the selector shall be clearly identifiable and shall exclusively enable one control or operating mode to be selected…
IEC 60204-1:2016	9.2.3.5 Operating modes ... The selector may be replaced by another selection method which restricts the use of certain functions of the machinery to certain categories of operator (for example access code). Mode selection by itself shall not initiate machine operation. A separate actuation of the start control shall be required. ...Indication of the selected operating mode shall be provided...
EN ISO 13849-1:2015	5.2.2 Manual reset function (See 5.6 Reset Behavior with ISO 13849-1:2015)

6.4.2 Interface Description

FB Name	SF_ModeSelector		
This function block selects the system operation mode, such as manual, automatic, semi-automatic, etc.			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
Activate	BOOL	FALSE	See Table 5.1.1 General Input Parameters
S_Mode0	SAFEBOOL	FALSE	Variable or constant. Input 0 from mode selector switch. FALSE: Mode 0 is not requested by operator. TRUE: Mode 0 is requested by operator.
S_Mode1	SAFEBOOL	FALSE	Variable or constant. Input 1 from mode selector switch. FALSE: Mode 1 is not requested by operator. TRUE: Mode 1 is requested by operator.
S_Mode2	SAFEBOOL	FALSE	Variable or constant. Input 2 from mode selector switch. FALSE: Mode 2 is not requested by operator. TRUE: Mode 2 is requested by operator.
S_Mode3	SAFEBOOL	FALSE	Variable or constant. Input 3 from mode selector switch. FALSE: Mode 3 is not requested by operator. TRUE: Mode 3 is requested by operator.
S_Mode4	SAFEBOOL	FALSE	Variable or constant. Input 4 from mode selector switch. FALSE: Mode 4 is not requested by operator. TRUE: Mode 4 is requested by operator.
S_Mode5	SAFEBOOL	FALSE	Variable or constant. Input 5 from mode selector switch. FALSE: Mode 5 is not requested by operator. TRUE: Mode 5 is requested by operator.
S_Mode6	SAFEBOOL	FALSE	Variable or constant. Input 6 from mode selector switch. FALSE: Mode 6 is not requested by operator. TRUE: Mode 6 is requested by operator.

S_Mode7	SAFEBOOL	FALSE	Variable or constant. Input 7 from mode selector switch. FALSE: Mode 7 is not requested by operator. TRUE: Mode 7 is requested by operator.
S_Unlock	SAFEBOOL	FALSE	Variable or constant. Locks the selected mode. FALSE: The actual S_ModeXSel output is locked therefore a change of any S_ModeX input does **not** lead to a change in the S_ModeXSel output even in the event of a rising edge of SetMode. TRUE: The selected S_ModeXSel is not locked; a mode selection change is possible.
S_SetMode	SAFEBOOL	FALSE	Variable (or constant FALSE, if AutoSetMode = TRUE) Sets the selected mode. Operator acknowledges the setting of a mode. Any change to new S_ModeX = TRUE leads to S_AnyModeSel/S_ModeXSel = FALSE, only a rising SetMode trigger then leads to new S_ModeXSel = TRUE.
AutoSetMode	BOOL	FALSE	Constant. Parameterizes the acknowledgement mode. FALSE: A change in mode must be acknowledged by the operator via SetMode. TRUE: A valid change of the S_ModeX input to another S_ModeX automatically leads to a change in S_ModeXSel without operator acknowledgment via SetMode (as long as this is not locked by S_Unlock).
ModeMonitorTime	TIME	T#0	Constant. Maximum permissible time for changing the selection input.
Reset	BOOL	FALSE	See Section 5.1.1 General Input Parameters
VAR_OUTPUT			
Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
S_Mode0Sel	SAFEBOOL	FALSE	Indicates that mode 0 is selected and acknowledged. FALSE: Mode 0 is not selected or not active. TRUE: Mode 0 is selected and active.
S_Mode1Sel	SAFEBOOL	FALSE	Indicates that mode 1 is selected and acknowledged. FALSE: Mode 1 is not selected or not active. TRUE: Mode 1 is selected and active.
S_Mode2Sel	SAFEBOOL	FALSE	Indicates that mode 2 is selected and acknowledged. FALSE: Mode 2 is not selected or not active. TRUE: Mode 2 is selected and active.
S_Mode3Sel	SAFEBOOL	FALSE	Indicates that mode 3 is selected and acknowledged. FALSE: Mode 3 is not selected or not active. TRUE: Mode 3 is selected and active.
S_Mode4Sel	SAFEBOOL	FALSE	Indicates that mode 4 is selected and acknowledged. FALSE: Mode 4 is not selected or not active. TRUE: Mode 4 is selected and active.
S_Mode5Sel	SAFEBOOL	FALSE	Indicates that mode 5 is selected and acknowledged. FALSE: Mode 5 is not selected or not active. TRUE: Mode 5 is selected and active.
S_Mode6Sel	SAFEBOOL	FALSE	Indicates that mode 6 is selected and acknowledged. FALSE: Mode 6 is not selected or not active. TRUE: Mode 6 is selected and active.
S_Mode7Sel	SAFEBOOL	FALSE	Indicates that mode 7 is selected and acknowledged. FALSE: Mode 7 is not selected or not active. TRUE: Mode 7 is selected and active.
S_AnyModeSel	SAFEBOOL	FALSE	Indicates that any of the 8 modes is selected and acknowledged. FALSE: No S_ModeX is selected. TRUE: One of the 8 S_ModeX is selected and active.
SafetyDemand	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters

ResetRequest	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters
Notes: The X in parameter names "S_ModeX" or "S_ModeXSel" is a placeholder for digits 0 to 7.			



6.4.3 Functional Description

This function block selects the system operation mode, such as manual, automatic, semi-automatic, etc. On controller startup, it should be assumed that the machine is in safe mode. On machine startup, the transition to the mode set by the mode selector switch must be initiated by a function block input (e.g., machine START button).

The default state following activation of the FB is the ModeChanged state. This is also the safe state of the FB, where all S_ModeXSel and S_AnyModeSel are FALSE.

If the FB is in the ModeChanged state:

- The new S_ModeX input must be acknowledged by a rising S_SetMode trigger (if AutoSetMode = FALSE), which leads to a new S_ModeXSel output.
- The new S_ModeX input automatically leads to a new S_ModeXSel output (if AutoSetMode = TRUE).
- Such a transition from state 8802 to 8000 is only valid, if one S_ModeX input is TRUE. As long as all S_ModeX are FALSE, the FB remains in state 8802, even if the S_SetMode triggers.

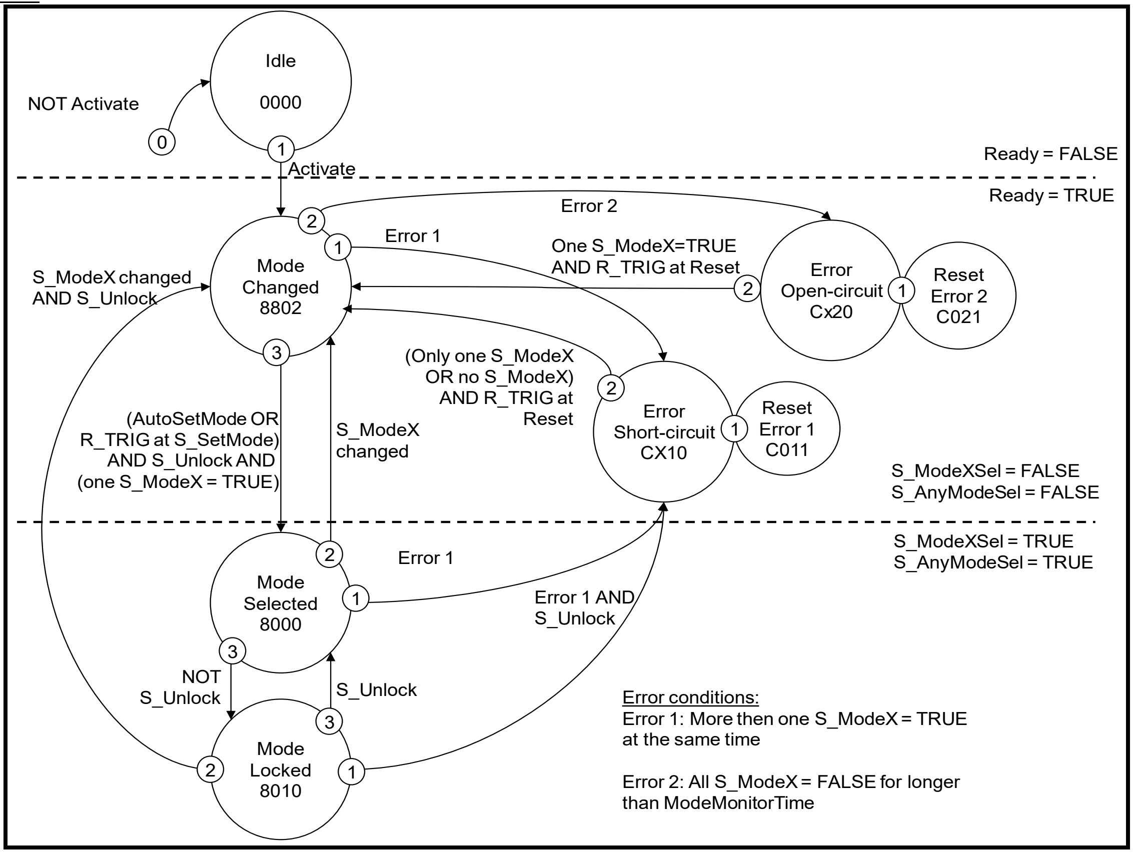
The transition from the ModeChanged to ModeSelected state, i.e., S_SetMode set by the operator, is not monitored by a timer.

If the FB is in the ModeSelected state, the simultaneous occurrence of a new S_ModeX input (higher priority) and the NOT S_Unlock signal (lower priority) leads to the ModeChanged state.

The S_ModeX input parameters, which are not used for mode selection, should be called with the default value FALSE to simplify program verification.

The AutoSetMode input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

State Diagram



Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Figure 21: State diagram for SF_ModeSelector

Typical Timing Diagrams

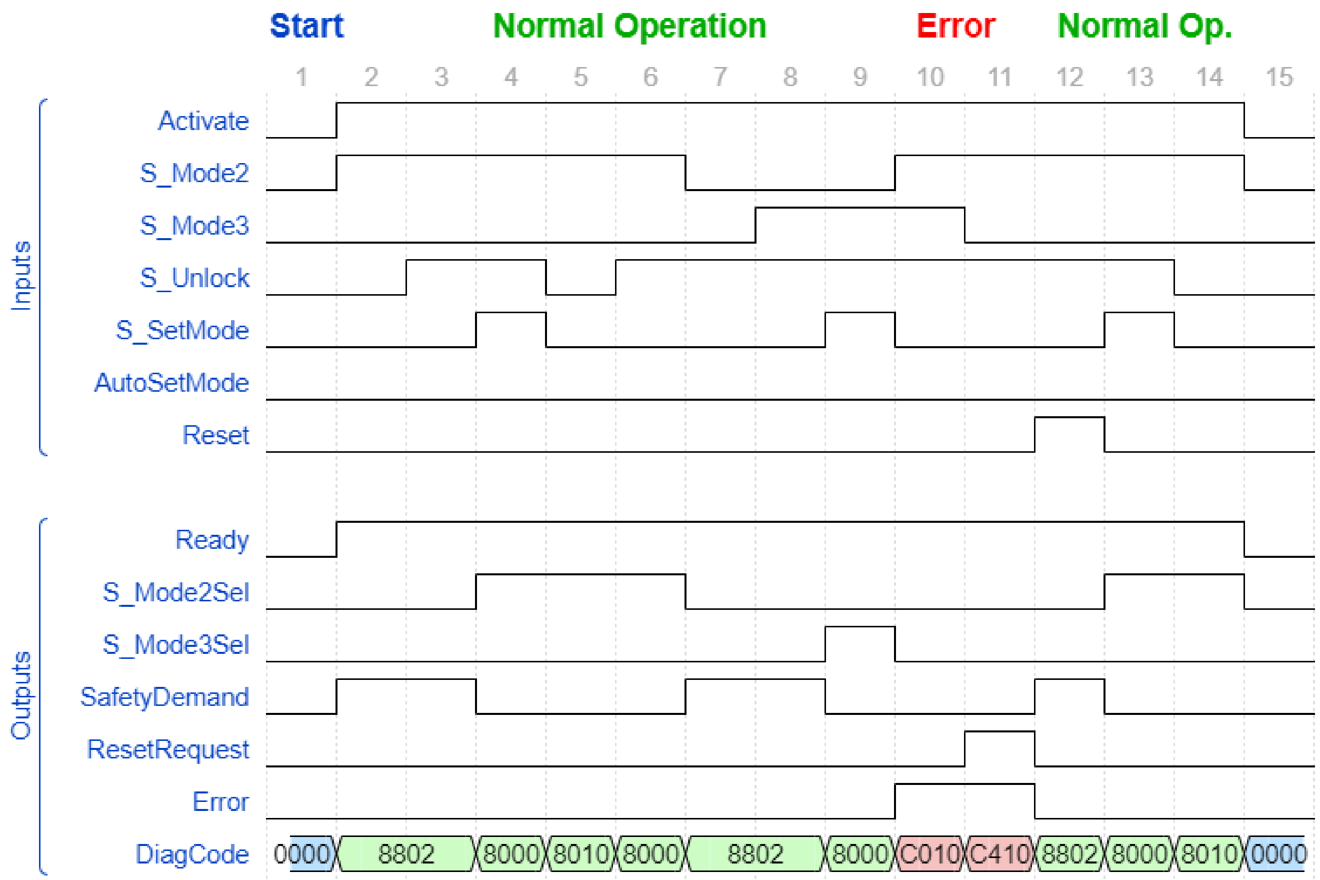


Figure 22: Timing diagram for SF_ModeSelector, valid change in Mode input with acknowledgment

6.4.4 Error Detection

The FB detects if none of the mode inputs is selected. This invalid condition is detected after ModeMonitorTime has elapsed. ModeMonitorTime restarts with each falling trigger of an S_ModeX switched mode input, and the state transfers to ModeChanged following the activation of the FB
In contrast, the FB directly detects whether more than one S_ModeX mode input is selected at the same time.
A static reset condition is detected when the FB is either in Error state C011 or C021.

6.4.5 Error Behavior

In the event of an error, the S_ModeXSel and S_AnyModeSel outputs are set to safe state FALSE. The DiagCode output indicates the relevant error code, and the Error output is set to TRUE.
An error must be acknowledged with the rising trigger of the Reset BOOL input. The FB changes from an error state to the ModeChanged state.

6.4.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

Cx10	Error
	Short-circuit
		The FB detected that two or more S_ModeX are TRUE, e.g., short-circuit of cables.
		IF (Only one S_ModeX OR no S_ModeX) = TRUE THEN x = 4 ELSE x = 0
		
		Output signals for x = 4 (C410):
		Ready = TRUE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE
		S_AnyModeSel = FALSE
		All S_ModeXSel = FALSE
		
		Output signals for x = 0 (C010):
		Ready = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE
		S_AnyModeSel = FALSE
		All S_ModeXSel = FALSE

Cx20	Error
	Open-circuit
		The FB detected that all S_ModeX are FALSE: The period following a falling S_ModeX trigger exceeds ModeMonitorTime, e.g., open-circuit of cables.
		
		IF (Only one S_ModeX) = TRUE THEN x = 4 ELSE x = 0
		
		Output signals for x = 4 (C420):
		Ready = TRUE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE
		S_AnyModeSel = FALSE
		All S_ModeXSel = FALSE
		
		Output signals for x = 0 (C020):
		Ready = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE
		S_AnyModeSel = FALSE
		All S_ModeXSel = FALSE

C011	Reset Error 1	Static Reset signal detected in state C410.
		Ready = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE
		S_AnyModeSel = FALSE
		All S_ModeXSel = FALSE

C021	Reset Error 2	Static Reset signal detected in state C420.
		Ready = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE
		S_AnyModeSel = FALSE
		All S_ModeXSel = FALSE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE
		S_AnyModeSel = FALSE
		All S_ModeXSel = FALSE

8802	ModeChanged	State after activation or when S_ModeX has changed (unless locked) or after Reset of an error state.
		Ready = TRUE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE
		S_AnyModeSel = FALSE
		All S_ModeXSel = FALSE

8000	ModeSelected	Valid mode selection, but not yet locked.
		Ready = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE
		S_AnyModeSel = TRUE
		S_ModeXSel = Selected X is TRUE, others are FALSE.

8010	ModeLocked	Valid mode selection is locked.
		Ready = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE
		S_AnyModeSel = TRUE
		S_ModeXSel = Selected X is TRUE, others are FALSE.

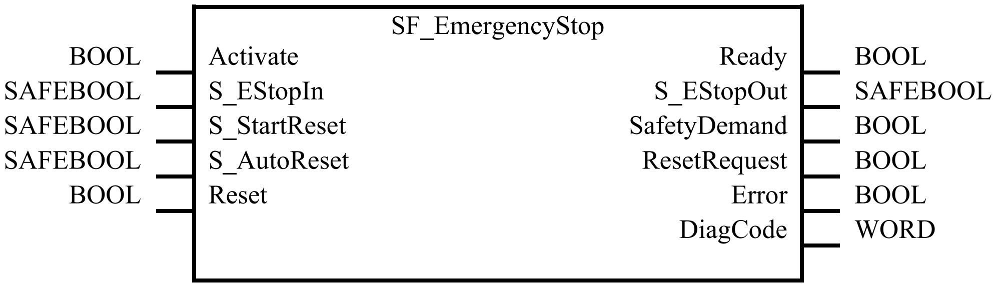
6.5 Emergency Stop

6.5.1 Applicable Safety Standards

Standards	**Requirements**
IEC 60204-1:2016	9.2.3.4 Emergency operations (emergency stop, emergency switching off) … The reset of the command shall not restart the machinery but only permit restarting.
EN ISO 13849-1:2015	5.2.2 Manual reset function
EN ISO 12100: 2010	6.2.11.4: Restart following power failure/spontaneous restart
EN ISO 13850:2015	Emergency Stop

6.5.2 Interface Description

FB Name	**SF_EmergencyStop**			
This function block is a safety-related function block for monitoring an emergency stop button. This FB can be used for emergency switch off functionality (stop category 0), or - with additional peripheral support - as emergency stop (stop category 1 or 2)				
VAR_INPUT				
	Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
	Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
	S_EStopIn	SAFEBOOL	FALSE	Safety demand input. Variable. FALSE: Demand for safety-related response (e.g., emergency stop button is engaged). TRUE: No demand for safety-related response (e.g., emergency stop button not engaged).
	S_StartReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
	S_AutoReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
	Reset	BOOL	FALSE	See Section 5.1.1 General Input Parameters
VAR_OUTPUT				
	Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
	S_EStopOut	SAFEBOOL	FALSE	Output for the safety-related response. FALSE: Safety output disabled. Demand for safety-related response (e.g., emergency stop button engaged, reset required or internal errors active) TRUE: Safety output enabled. No demand for safety-related response (e.g., emergency stop button not engaged, no internal errors active).
	SafetyDemand	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
	ResetRequest	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
	Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
	DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters
Notes: -				



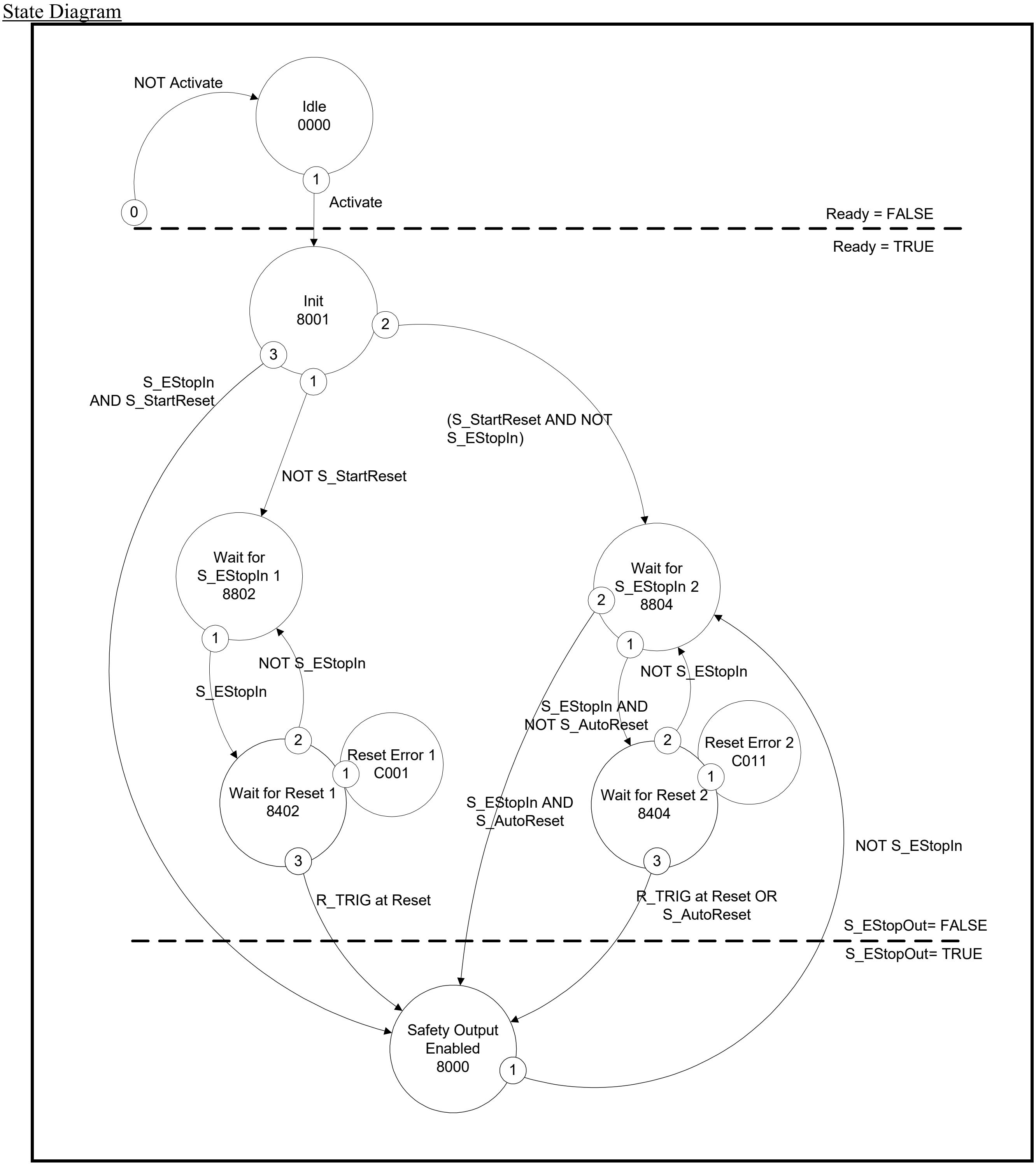
6.5.3 Functional Description

The S_EStopOut enable signal is reset to FALSE as soon as the S_EStopIn input is set to FALSE. The S_EStopOut enable signal is reset to TRUE only if the S_EStopIn input is set to TRUE, and a reset occurs. The enable reset depends on the defined S_StartReset, S_AutoReset, and Reset inputs.
If S_AutoReset = TRUE, acknowledgment is automatic.
If S_AutoReset = FALSE, a rising trigger at the Reset input must be used to acknowledge the enable.
If S_StartReset = TRUE, acknowledgment is automatic the first time the PES is started.
If S_StartReset = FALSE, a rising trigger at the Reset input must be used to acknowledge the enable.

The S_StartReset and S_AutoReset inputs shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

SF_EmergencyStop can be used to monitor both single and two-channel emergency stop buttons. For example, for two-channel applications, the additional function blocks SF_Equivalent can be used to detect whether the contact synchronization has been exceeded. The category classification in accordance with EN ISO 13849-1 will depend on the final elements that are used.
The SF_EmergencyStop automatically detects a static TRUE on Reset. Further error detection, e.g., wire break, short circuit depends on the dedicated hardware that is used.

State Diagram



Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Figure 23: State diagram for SF_EmergencyStop

Typical Timing Diagrams

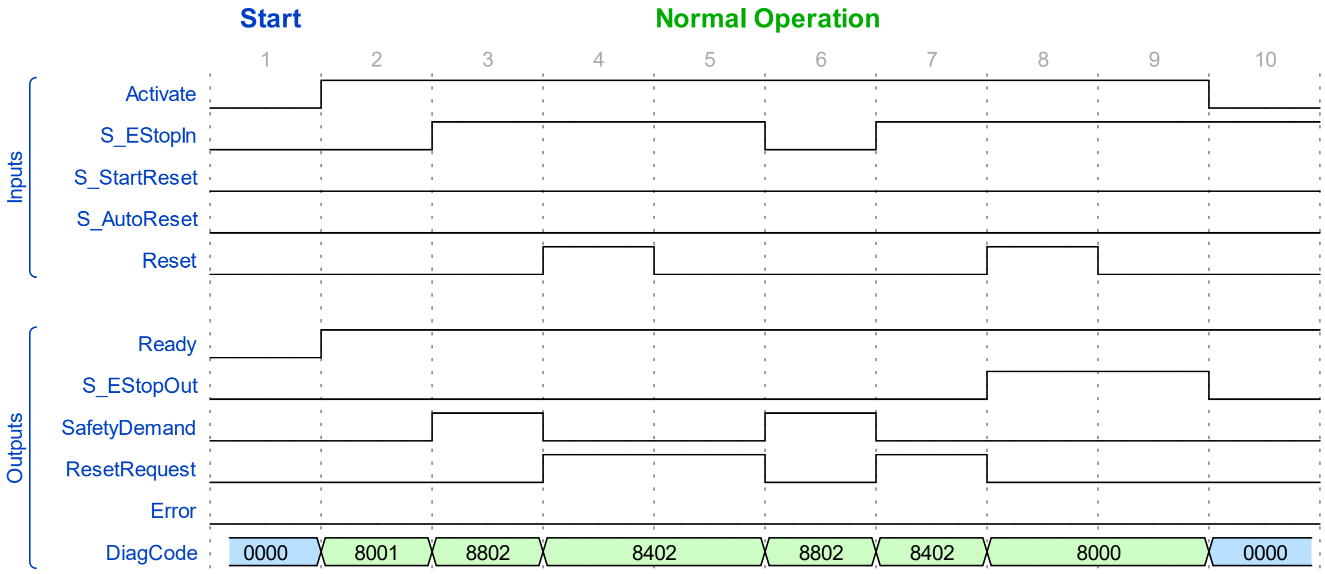


Figure 24: Timing diagram for SF_EmergencyStop: S_StartReset = FALSE

Figure 25: Timing diagram for SF_EmergencyStop: S_StartReset = TRUE

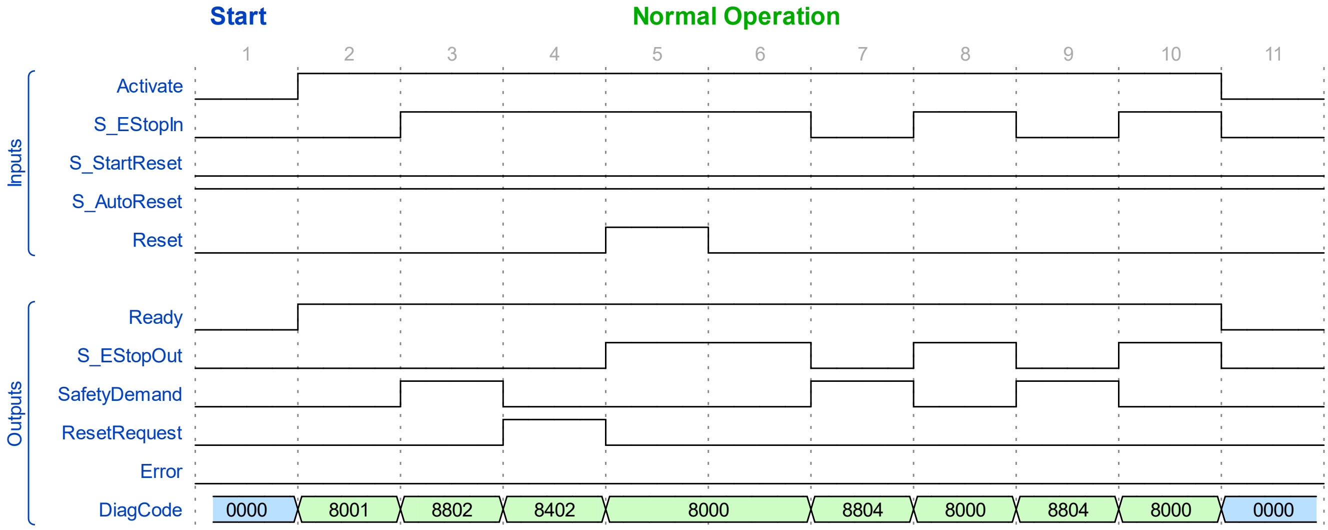


Figure 26: Timing diagram for SF_EmergencyStop: S_StartReset = FALSE, S_AutoReset = TRUE

6.5.4 Error Detection

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

6.5.5 Error Behavior

S_EStopOut is set to FALSE. In case of a static TRUE signal at the Reset input, the DiagCode output indicates the relevant error code and the Error output is set to TRUE.
To leave the error states, the Reset must be set to FALSE.

6.5.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C001	Reset Error 1	Reset is TRUE while waiting for S_EStopIn = TRUE.
		Ready = TRUE
		S_EStopOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C011	Reset Error 2	Reset is TRUE while waiting for S_EStopIn = TRUE.
		Ready = TRUE
		S_EStopOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_EStopOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8001	Init	Activation is TRUE. The function block was enabled. Check if S_StartReset is required.
		Ready = TRUE
		S_EStopOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8802	Wait for S_EstopIn 1	Activation is TRUE. Check if Reset is FALSE and wait for S_EStopIn = TRUE.
		Ready = TRUE
		S_EStopOut = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8402	Wait for Reset 1	Activation is TRUE. S_EStopIn = TRUE. Wait for rising trigger of Reset.
		Ready = TRUE
		S_EStopOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = TRUE
		Error = FALSE

8804	Wait for S_EstopIn 2	Activation is TRUE. Safety demand detected. Check if Reset is FALSE and wait for S_EStopIn = TRUE.
		Ready = TRUE
		S_EStopOut = FALSE
		SafetyDemand = TRUE
		ResetRequest = NOT Reset
		Error = FALSE

8404	Wait for Reset 2	Activation is TRUE. S_EStopIn = TRUE. Check for S_AutoReset or wait for rising trigger of Reset.
		Ready = TRUE
		S_EStopOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8000	Safety Output Enabled	Activation is TRUE. S_EStopIn = TRUE. Functional mode with S_EStopOut = TRUE.
		Ready = TRUE
		S_EStopOut = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

6.6 Electro-Sensitive Protective Equipment (ESPE)

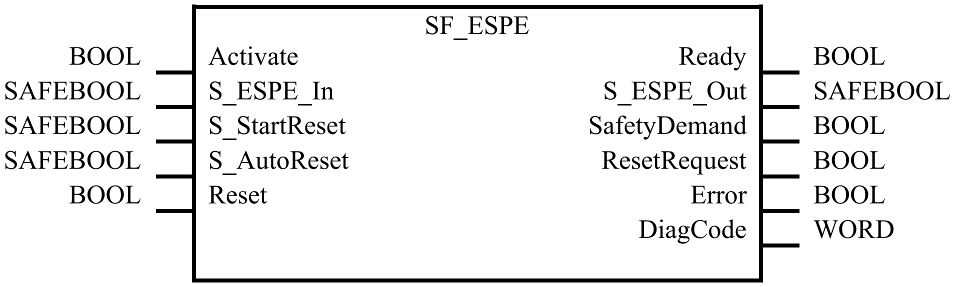
Basically, this function block is functionally equivalent to 6.5 Emergency Stop except for the dedicated input and output names and the applicable safety standards.

6.6.1 Applicable Safety Standards

Standards	**Requirements**
IEC 61496-1: 2012	A.5.1 Start Interlock: The start interlock shall prevent the OSSD(s) going to the ON-state when the electrical supply is switched on, or is interrupted and restored. A.5.2: A failure of the start interlock which causes it to go to, or remain in a permanent ON-state shall cause the ESPE to go to, or to remain in the lock-out condition. A.6.1 Restart interlock: … The interlock condition shall continue until the restart interlock is manually reset. However, it shall not be possible to reset the restart interlock whilst the sensing device is actuated.
EN ISO 13849-1:2015	5.2.2 Manual reset function
ISO 12100-2: 2010	4.11.4: Restart following power failure/spontaneous restart

6.6.2 Interface Description

FB Name	**SF_ESPE**		
This function block is a safety-related function block for monitoring electro-sensitive protective equipment (ESPE).			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
S_ESPE_In	SAFEBOOL	FALSE	Safety demand input. Variable. FALSE: ESPE actuated, demand for safety-related response. TRUE: ESPE not actuated, no demand for safety-related response. Safety control system must be able to detect a very short interruption of the sensor (which is specified in 61496-1: minimum 80 ms), when the ESPE is used in applications as a trip device
S_StartReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
S_AutoReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
Reset	BOOL	FALSE	See Section 5.1.1 General Input Parameters
VAR_OUTPUT			
Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
S_ESPE_Out	SAFEBOOL	FALSE	Output for the safety-related response. FALSE: Safety output disabled. Demand for safety-related response (e.g., reset required or internal errors active). TRUE: Safety output enabled. No demand for safety-related response.
SafetyDemand	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
ResetRequest	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters
Notes:			



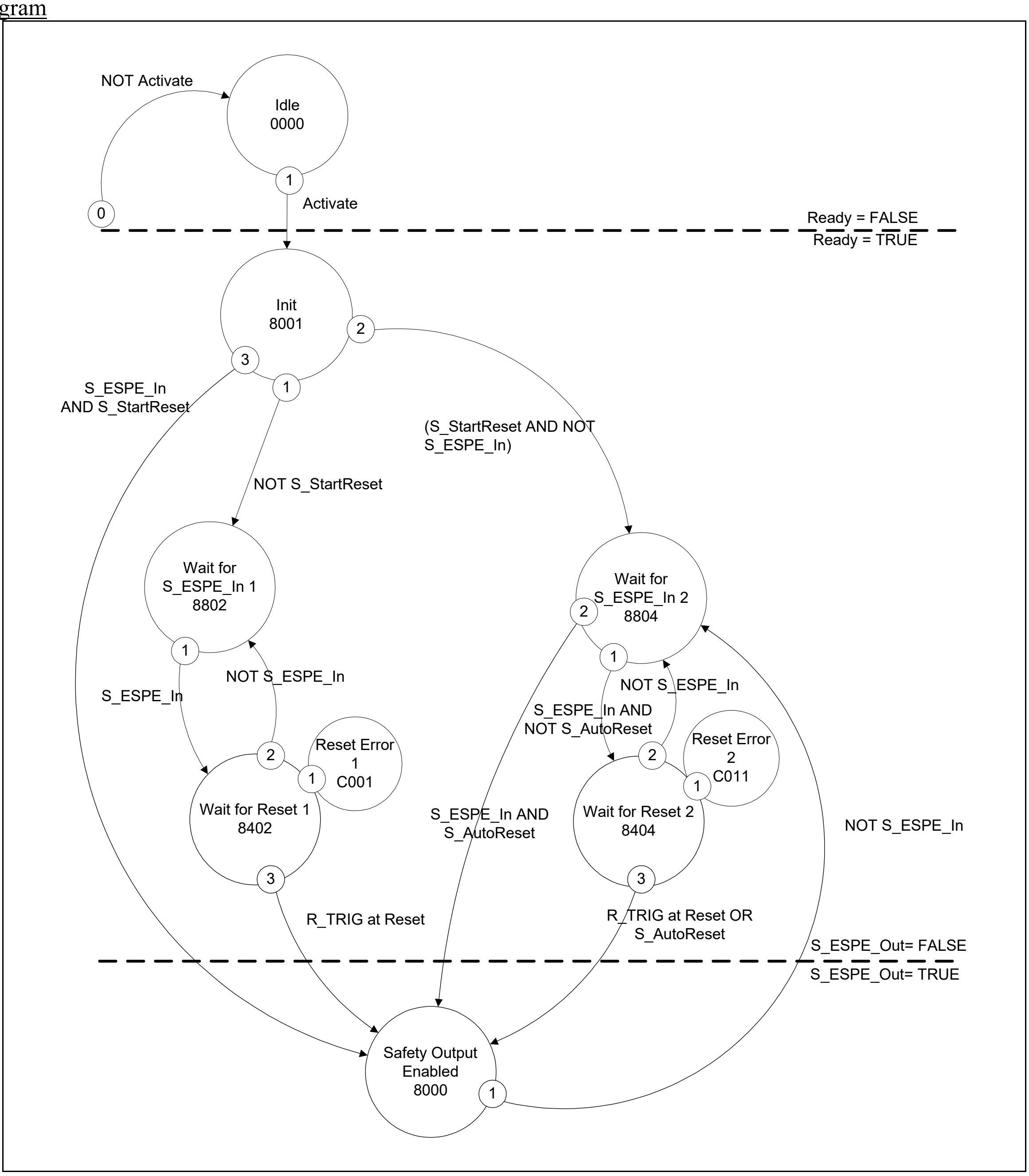
6.6.3 Functional Description

This function block is a safety-related function block for monitoring electro-sensitive protective equipment (ESPE). The function is identical to SF_EmergencyStop. The S_ESPE_Out output signal is set to FALSE as soon as the S_ESPE_In input is set to FALSE. The S_ESPE_Out output signal is set to TRUE only if the S_ESPE_In input is set to TRUE and a reset occurs. The enable reset depends on the defined S_StartReset, S_AutoReset, and Reset inputs.
If S_AutoReset = TRUE, acknowledgment is automatic.
If S_AutoReset = FALSE, a rising trigger at the Reset input must be used to acknowledge the enable.
If S_StartReset = TRUE, acknowledgment is automatic the PES is started the first time.
If S_StartReset = FALSE, a rising trigger at the Reset input must be used to acknowledge the enable.

The S_StartReset and S_AutoReset inputs shall only be activated if it is ensured, that no hazardous situation can occur when the PES is started.

The ESPE must be selected in respect of the product standards EN IEC 61496-1, -2 and -3 and the required categories according EN 954-1.

State Diagram



Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Figure 27: State diagram for SF_ESPE

Typical Timing Diagrams

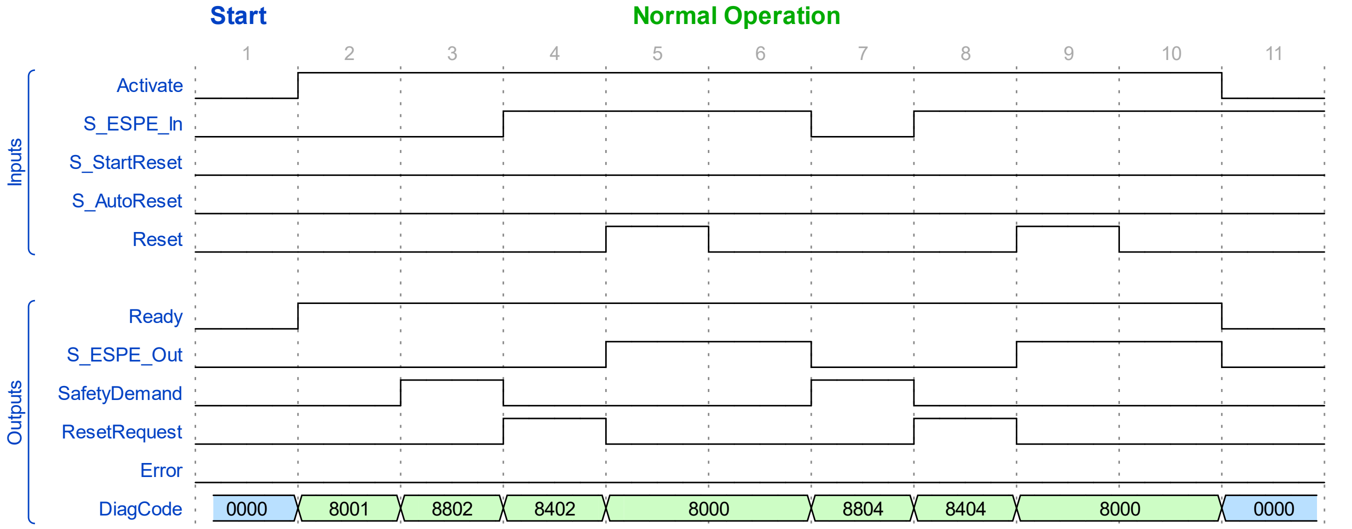


Figure 28: Timing diagram for SF_ESPE: S_StartReset = FALSE

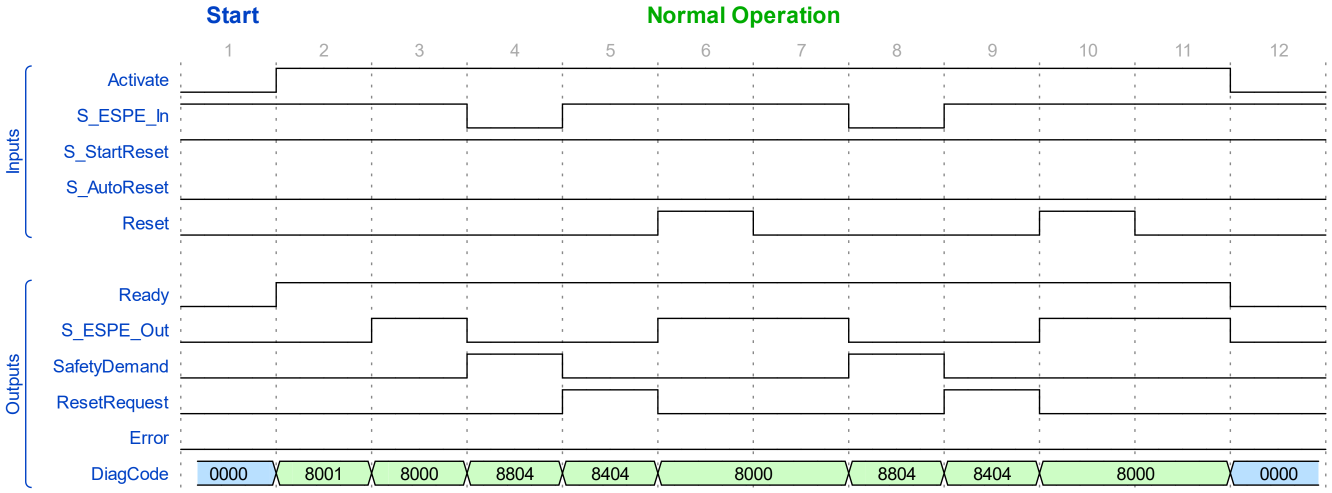


Figure 29:TimingdiagramforSF_ESPE:S_StartReset=TRUE

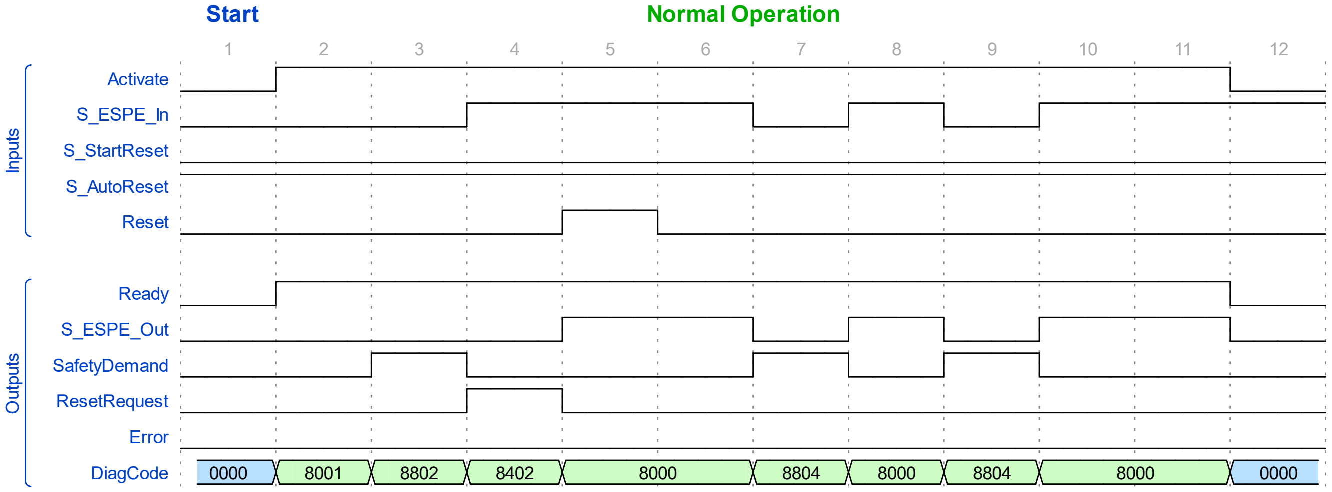


Figure 30: Timing diagram for SF_ESPE: S_StartReset = FALSE, S_AutoReset = TRUE

6.6.4 Error Detection

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

6.6.5 Error Behavior

S_ESPE_Out is set to FALSE. In case of a static TRUE signal at the Reset input, the DiagCode output indicates the relevant error code, and the Error output is set to TRUE.
To leave the error states, the Reset must be set to FALSE.

6.6.6 Function Block-Specific Error and Status Codes

B-specific error codes:

DiagCode	State Name	State Description and Output Setting

C001	Reset Error 1	Reset is TRUE while waiting for S_ESPE_In = TRUE.
		Ready = TRUE
		S_ESPE_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C011	Reset Error 2	Reset is TRUE while waiting for S_ESPE_In = TRUE.
		Ready = TRUE
		S_ESPE_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_ESPE_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8001	Init	Activation is TRUE. The function block was enabled. Check if S_StartReset is required.
		Ready = TRUE
		S_ESPE_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8802	Wait for S_ESPE_In 1	Activation is TRUE. Check if Reset is FALSE and wait for S_ESPE_In = TRUE.
		Ready = TRUE
		S_ESPE_Out = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8402	Wait for Reset 1	Activation is TRUE. S_ESPE_In = TRUE. Wait for rising trigger of Reset.
		Ready = TRUE
		S_ESPE_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8804	Wait for S_ESPE_In 2	Activation is TRUE. Safety demand detected. Check if Reset is FALSE and wait for S_ESPE_In = TRUE.
		Ready = TRUE
		S_ESPE_Out = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8404	Wait for Reset 2	Activation is TRUE. S_ESPE_In = TRUE. Check for S_AutoReset or wait for rising trigger of Reset.
		Ready = TRUE
		S_ESPE_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

8000	Safety Output Enabled	Activation is TRUE. S_ESPE_In = TRUE. Functional mode with S_ESPE_Out = TRUE.
		Ready = TRUE
		S_ESPE_Out = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

6.7 Pressure Sensitive Equipment (PSE)

Basically, this function block is functionally equivalent to 6.5 Emergency Stop except for the dedicated input and output names and the applicable safety standards.

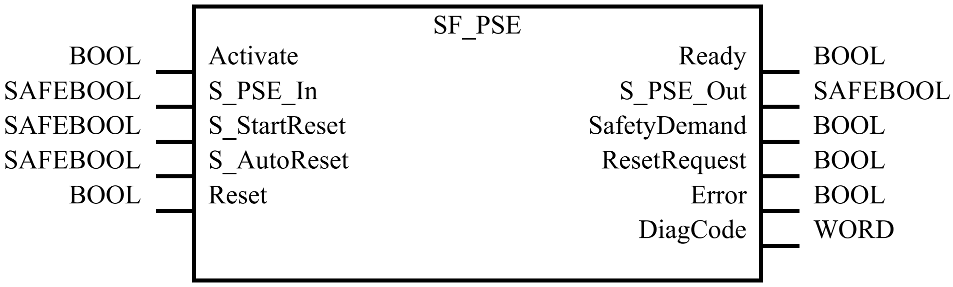
6.7.1 Applicable Safety Standards

Standards	**Requirements**
ISO 13856-1: 2013	Pressure-sensitive protective devices Part 1: General principles for the design and testing of pressure-sensitive mats and pressure-sensitive floors. 4.7 Response of output signal switching device(s) to the actuating force
ISO 13856-2:2013	Pressure-sensitive protective devices Part 2: General principles for the design and testing of pressure-sensitive edges and pressure-sensitive bars 4.11 Reset function
ISO 13856-3:2013	Pressure-sensitive protective devices Part 3: General principles for design and testing of pressure-sensitive bumpers, plates, wires and similar devices 4.2.6.3 Reset function. C.2.8 Result of sensor actuation
EN ISO 13849-1:2015	5.2.2 Manual reset function. <Note: a positive edge evaluation has the same quality as a negative edge evaluation>
ISO 12100-2: 2010	6.2.11.4 Restart after power interruption If a hazard could be generated, the spontaneous restart of a machine when it is re-energized after power interruption shall be prevented (for example, by use of a self-maintained relay, contactor or valve).

6.7.2 Interface Description

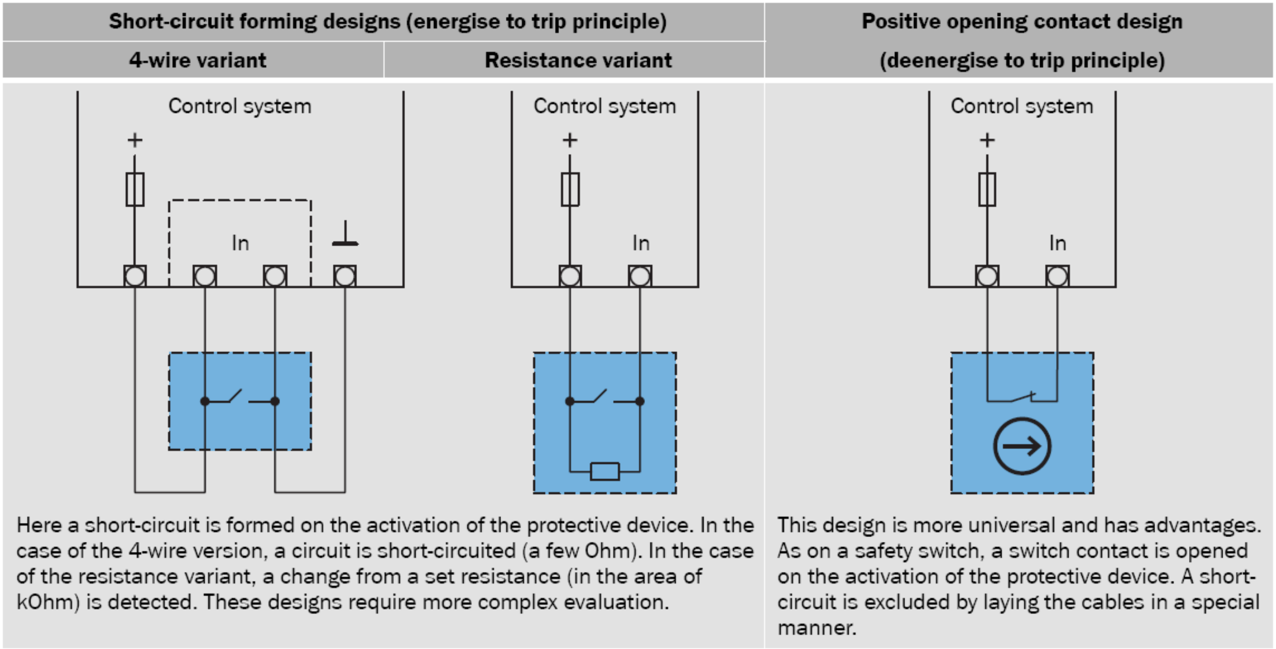
FB Name	**SF_PSE**		
This function block is a safety-related function block for monitoring Pressure-Sensitive-Equipment (PSE) like Safety Mats, Bumper etc.			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
S_PSE_In	SAFEBOOL	FALSE	Safety demand input. Variable. FALSE: PSE actuated, demand for safety-related response. TRUE: PSE not actuated, no demand for safety-related response. Safety control system must be able to detect a very short interruption of the PSE (which is specified in EN 1760: minimum 200 ms), when the PSE is used in applications as a safety device.
S_StartReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
S_AutoReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
Reset	BOOL	FALSE	See Section 5.1.1 General Input Parameters
VAR_OUTPUT			
Ready	BOOL	FALSE	See 5.1.2 General Output Parameters
S_PSE_Out	SAFEBOOL	FALSE	Output for the safety-related response. FALSE: Safety output disabled. Demand for safety-related response (e.g., reset requested or internal errors active). TRUE: Safety output enabled. No demand for safety-related response.
SafetyDemand	BOOL	FALSE	See 5.1.2 General Output Parameters
ResetRequest	BOOL	FALSE	See 5.1.2 General Output Parameters
Error	BOOL	FALSE	See 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See 5.1.2 General Output Parameters

Notes:



6.7.3 Functional Description

This function block is a safety-related function block for monitoring Pressure-Sensitive-Equipment (PSE) like Safety Mats, Bumper etc.


Short-circuit forming designs (energise to trip principle)
4-wire variant
Resistance variant
Positive opening contact design
(deenergise to trip principle)
Control system
+
In
Control system
+
In
Control system
+
In

Here a short-circuit is formed on the activation of the protective device. In the case of the 4-wire version, a circuit is short-circuited (a few Ohm). In the case of the resistance variant, a change from a set resistance (in the area of kOhm) is detected. These designs require more complex evaluation.

This design is more universal and has advantages. As on a safety switch, a switch contact is opened on the activation of the protective device. A short-circuit is excluded by laying the cables in a special manner.

Picture courtesy Sick AG

Figure 31: Overview of different configurations used in practice for PSE's

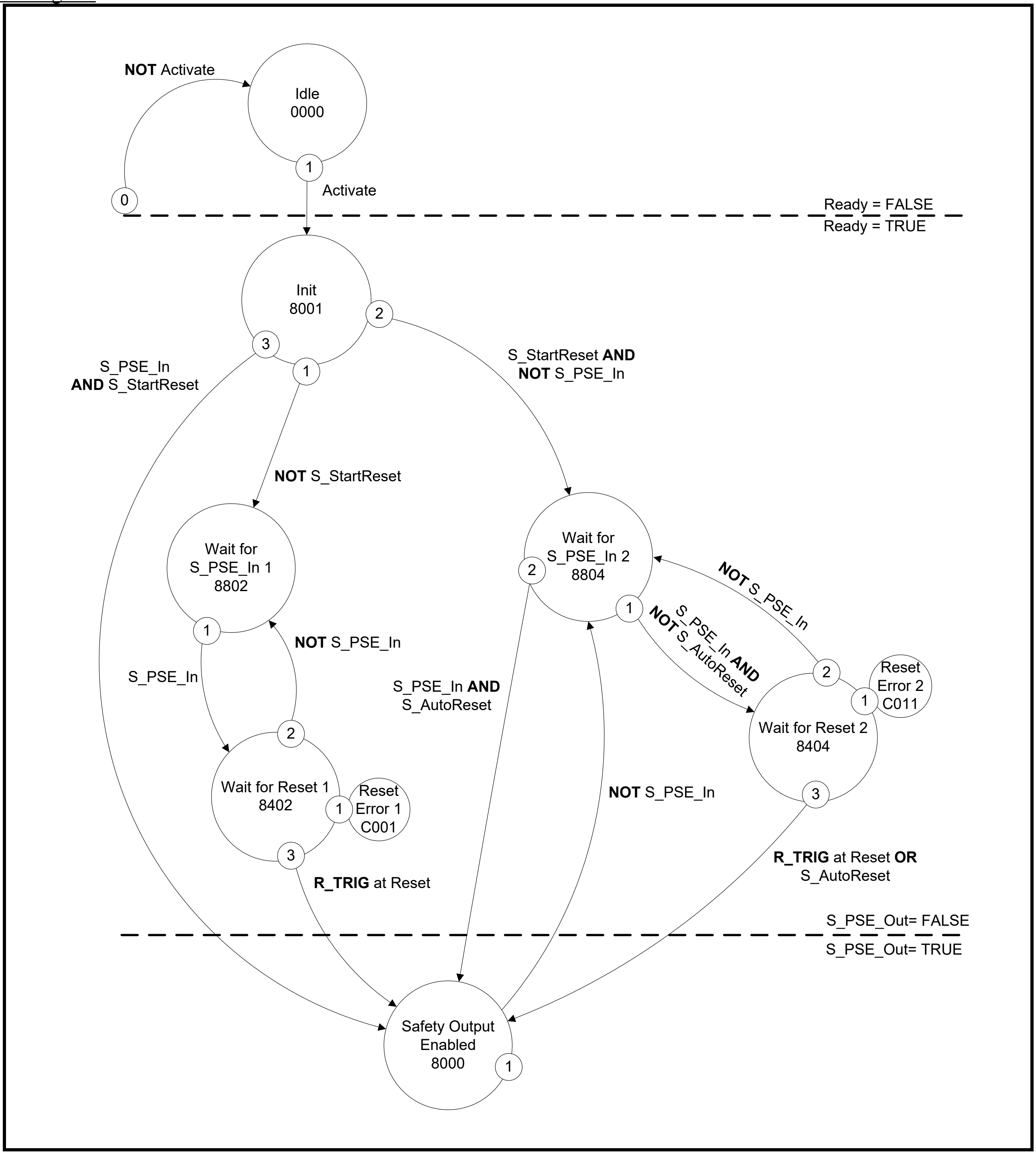
The Function Block requires a FALSE signal to activate the safety function. Therefore, a PSE with positive opening contact design, as shown in the figure above on the right side, can be connected directly to a safety input device. However, the other 2 principles as shown on the left require an evaluation unit to generate the applicable FALSE signal when the PSE is actuated.

The function is identical to SF_EmergencyStop (except for the 2 additional outputs SafetyDemand and ResetRequest). The S_PSE_Out output signal is set to FALSE as soon as the S_PSE_In input is set to FALSE. The S_PSE_Out output signal is set to TRUE only if the S_PSE_In input is set to TRUE and a reset occurs. The enable reset depends on the defined S_StartReset, S_AutoReset, and Reset inputs.
If S_AutoReset = TRUE, acknowledgment is automatic.
If S_AutoReset = FALSE, a rising trigger at the Reset input must be used to acknowledge the enable.
If S_StartReset = TRUE, acknowledgment is automatic the PES is started the first time.
If S_StartReset = FALSE, a rising trigger at the Reset input must be used to acknowledge the enable.

The S_StartReset and S_AutoReset inputs shall only be activated if it is ensured, that no hazardous situation can occur when the PES is started.

The SF_PSE must be selected in respect of the product standards EN 1760-1, -2 and -3 and the requested performance level according to ISO 13849-1:2008.

State Diagram



Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Figure 32: State diagram for SF_PSE

Typical Timing Diagrams

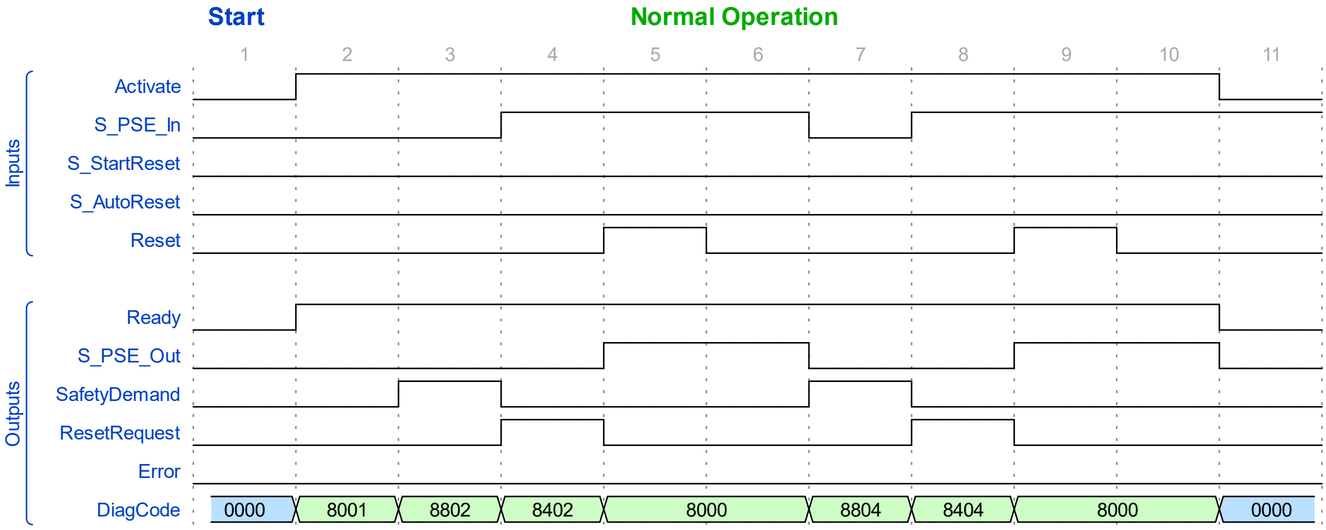


Figure 33: Timing diagram for SF_PSE: S_StartReset = FALSE; S_AutoReset = FALSE

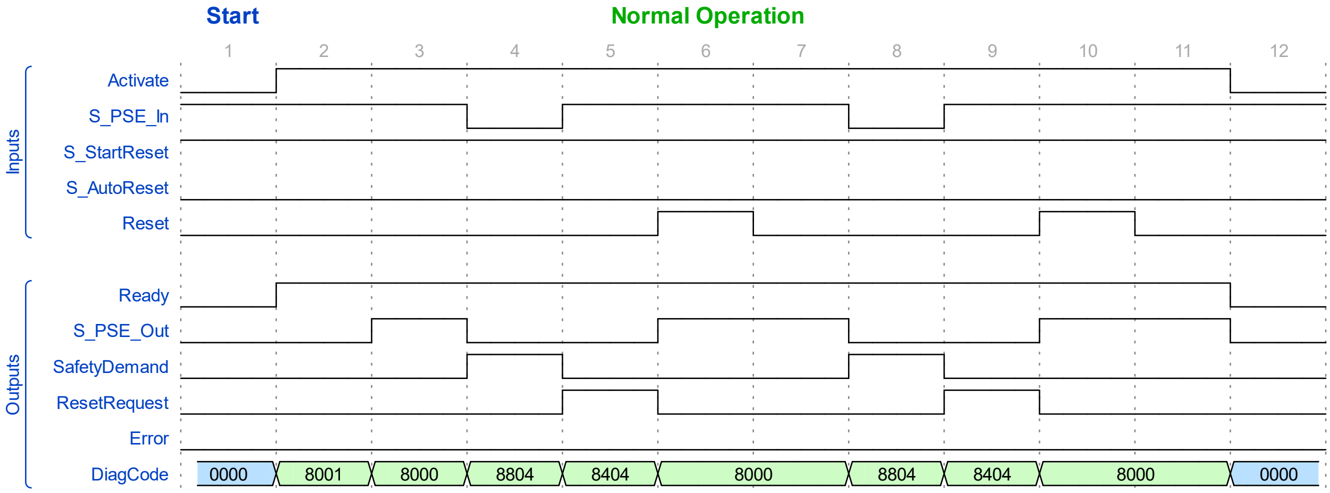


Figure 34: Timing diagram for SF_PSE: S_StartReset = TRUE, S_AutoReset = FALSE

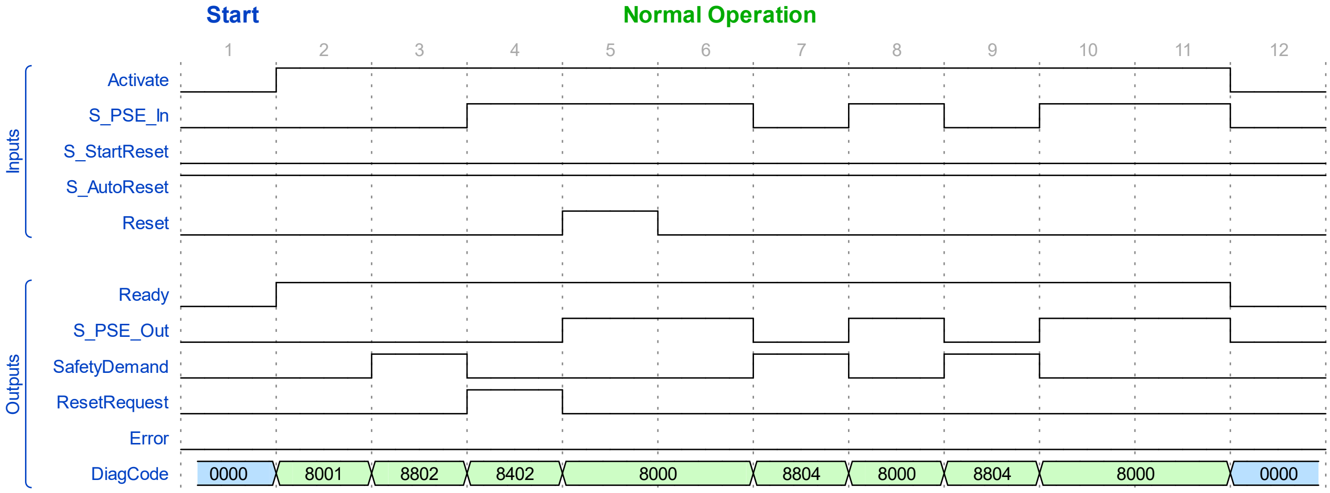


Figure 35: Timing diagram for SF_PSE: S_StartReset = FALSE, S_AutoReset = TRUE

6.7.4 Error Detection

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

6.7.5 Error Behavior

S_PSE_Out is set to FALSE. In case of a static TRUE signal at the Reset input, the DiagCode output indicates the relevant error code, and the Error output is set to TRUE.
To leave the error states, the Reset must be set to FALSE.

6.7.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C001	Reset Error 1	Reset is TRUE while waiting for S_PSE_In = TRUE.
		Ready = TRUE
		S_PSE_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C011	Reset Error 2	Reset is TRUE while waiting for S_PSE_In = TRUE.
		Ready = TRUE
		S_PSE_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_PSE_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8001	Init	Activation is TRUE. The function block was enabled. Check if S_StartReset is requested.
		Ready = TRUE
		S_PSE_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8802	Wait for S_PSE_In 1	Activation is TRUE. Check if Reset is FALSE and wait for S_PSE_In = TRUE.
		Ready = TRUE
		S_PSE_Out = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8402	Wait for Reset 1	Activation is TRUE. S_PSE_In = TRUE. Wait for rising trigger of Reset.
		Ready = TRUE
		S_PSE_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8804	Wait for S_PSE_In 2	Activation is TRUE. Safety demand detected. Check if Reset is FALSE and wait for S_PSE_In = TRUE.
		Ready = TRUE
		S_PSE_Out = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8404	Wait for Reset 2	Activation is TRUE. S_PSE_In = TRUE. Check for S_AutoReset or wait for rising trigger of Reset.
		Ready = TRUE
		S_PSE_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

8000	Safety Output Enabled	Activation is TRUE. S_PSE_In = TRUE. Functional mode with S_PSE_Out = TRUE.
		Ready = TRUE
		S_PSE_Out = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

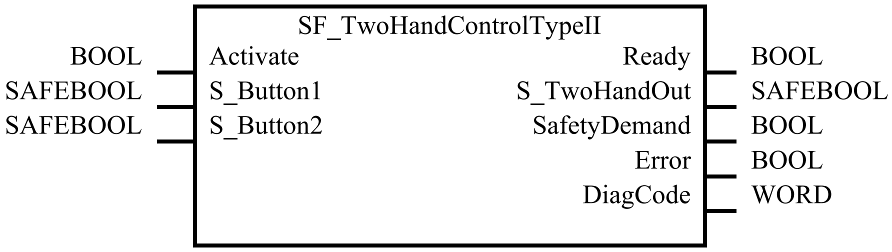
6.8 Two-Hand Control Type II

6.8.1 Applicable Safety Standards

Standards	Requirements
EN 574: 2008 ISO 13851:2002	Clause 4, Table 1, Type II. 5.1 Use of both hands / simultaneous actuation. 5.2 Relationship between output signal and input signals. 5.3 Completion of the output signal. 5.6 Reinitiation of the output signal. 6.3 Use of DIN EN 954-1 category 3
ISO 12100: 2010	6.2.11.4: Restart following power failure/spontaneous restart

6.8.2 Interface Description

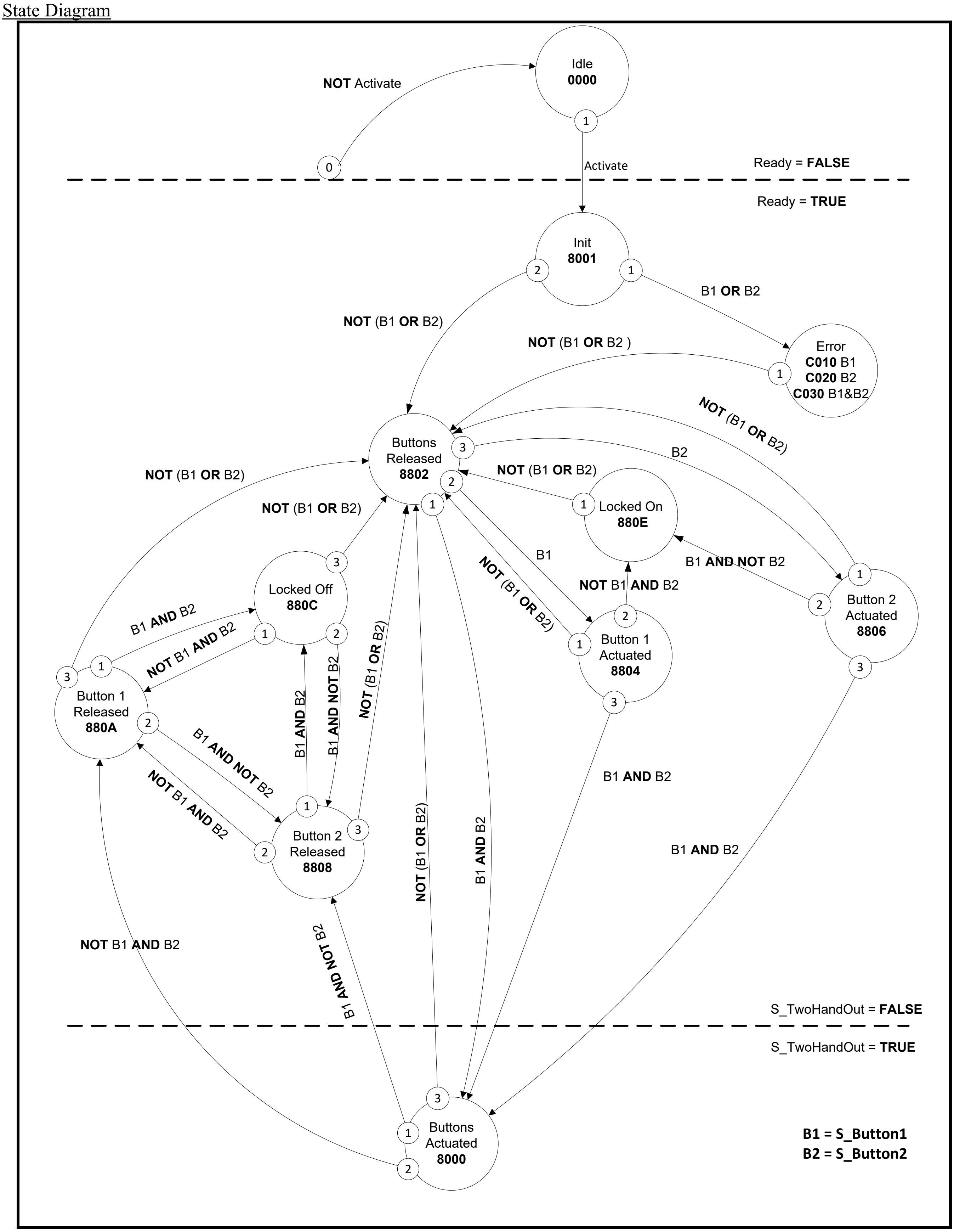
FB Name	**SF_TwoHandControlTypeII**		
This function block provides the two-hand control functionality (see EN 574, Section 4 Type II).			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
S_Button1	SAFEBOOL	FALSE	Variable. Input of button 1). FALSE: Button 1 released. TRUE: Button 1 actuated.
S_Button2	SAFEBOOL	FALSE	Variable. Input of button 2). FALSE: Button 2 released. TRUE: Button 2 actuated.
VAR_OUTPUT			
Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
S_TwoHandOut	SAFEBOOL	FALSE	Safety related output signal. FALSE: No correct two hand operation. TRUE: S_Button1 and S_Button2 inputs are TRUE, and no error occurred. Correct two hand operation.
SafetyDemand	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters
Notes: No Reset input or Error output is required, because no test can be performed on both switches.			



6.8.3 Functional Description

This function block provides the two-hand control functionality according to EN 574, Section 4 Type II. If S_Button1 and S_Button2 are set to TRUE in correct sequence, then the S_TwoHandOut output will also be set to TRUE. The FB also controls the release of both buttons before setting the output S_TwoHandOut again to TRUE.

State Diagram



Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Figure 36: State diagram for SF_TwoHandControlTypeII

Typical Timing Diagram

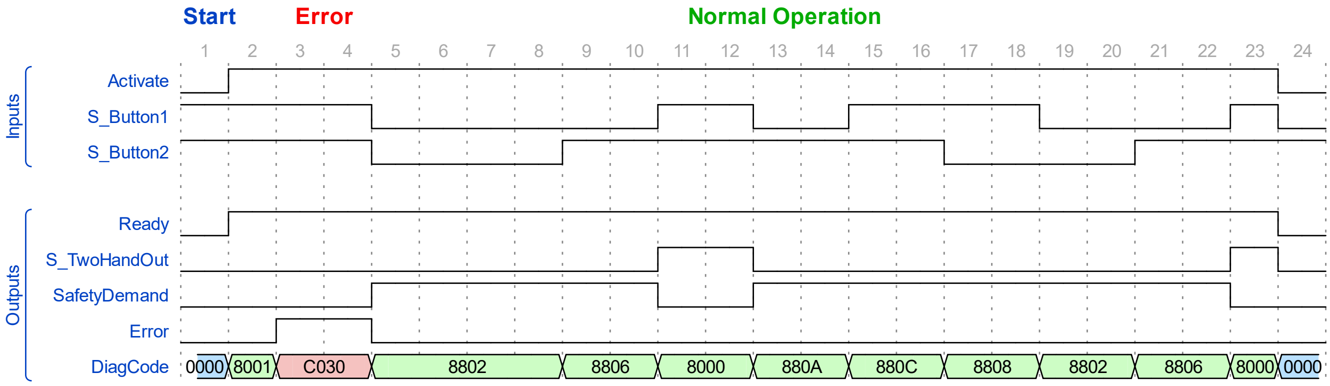


Figure 37: Timing diagram for SF_TwoHandControlTypeII

6.8.4 Error Detection

After activation of the FB, any button set to TRUE is detected as an invalid input setting leading to an error.

6.8.5 Error Behavior

In the event of an error, the S_TwoHandOut output is set to FALSE and remains in this safe state.
The Error state is left when both buttons are released (set to FALSE).

6.8.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C010	Error B1	S_Button1 was TRUE on FB activation.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = FALSE
		Error = TRUE

C020	Error B2	S_Button2 was TRUE on FB activation.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = FALSE
		Error = TRUE

C030	Error B1&B2	The signals at S_Button1 and S_Button2 were TRUE on FB activation.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_TwoHandOut = FALSE
		SafetyDemand = FALSE
		Error = FALSE

8000	Buttons Actuated	Both buttons actuated correctly. The safety related output is enabled.
		Ready = TRUE
		S_TwoHandOut = TRUE
		SafetyDemand = FALSE
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8001	Init	Function block is active, but in the Init state.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = FALSE
		Error = FALSE

8802	Buttons Released	No button is actuated.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

8804	Button 1 Actuated	Only Button 1 is actuated.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

8806	Button 2 Actuated	Only Button 2 is actuated.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

8808	Button 2 Released	The safety related output was enabled and is disabled again.
		FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output.
		In this state, S_Button1 is TRUE and S_Button2 is FALSE after disabling the safety related output.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

880A	Button 1 Released	The safety related output was enabled and is disabled again.
		FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output.
		In this state, S_Button1 is FALSE and S_Button2 is TRUE after disabling the safety related output.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

880C	Locked Off	The safety related output was enabled and is disabled again.
		FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output.
		In this state, S_Button1 is TRUE and S_Button2 is TRUE after disabling the safety related output.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

880E	Locked On	Incorrect actuation of the buttons. Waiting for release of both buttons.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

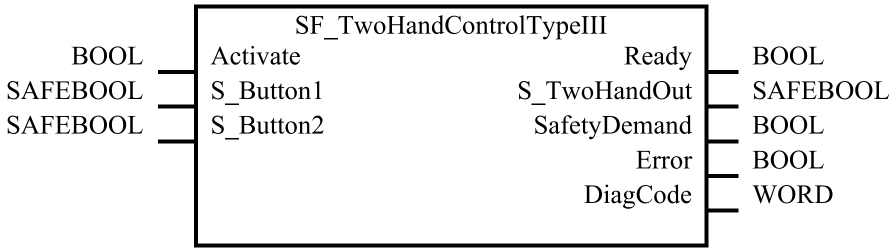
6.9 Two-Hand Control Type III

6.9.1 Applicable Safety Standards

Standards	**Requirements**
EN 574: 2008 ISO 13851:2002	Clause 4, Table 1, Type III A; B; C. 5.1 Use of both hands / simultaneous actuation. 5.2 Relationship between output signal and input signals. 5.3 Completion of the output signal. 5.6 Reinitiation of the output signal. 5.7 Synchronous actuation. 6.2 Use of EN 954-1 category 1. 6.3 Use of EN 954-1 category 3. 6.4 Use of EN 954-1 category 4. (Can only be realized by NO and NC switches together with antivalent processing)
ISO 12100: 2010	6.2.11.4: Restart following power failure/spontaneous restart

6.9.2 Interface Description

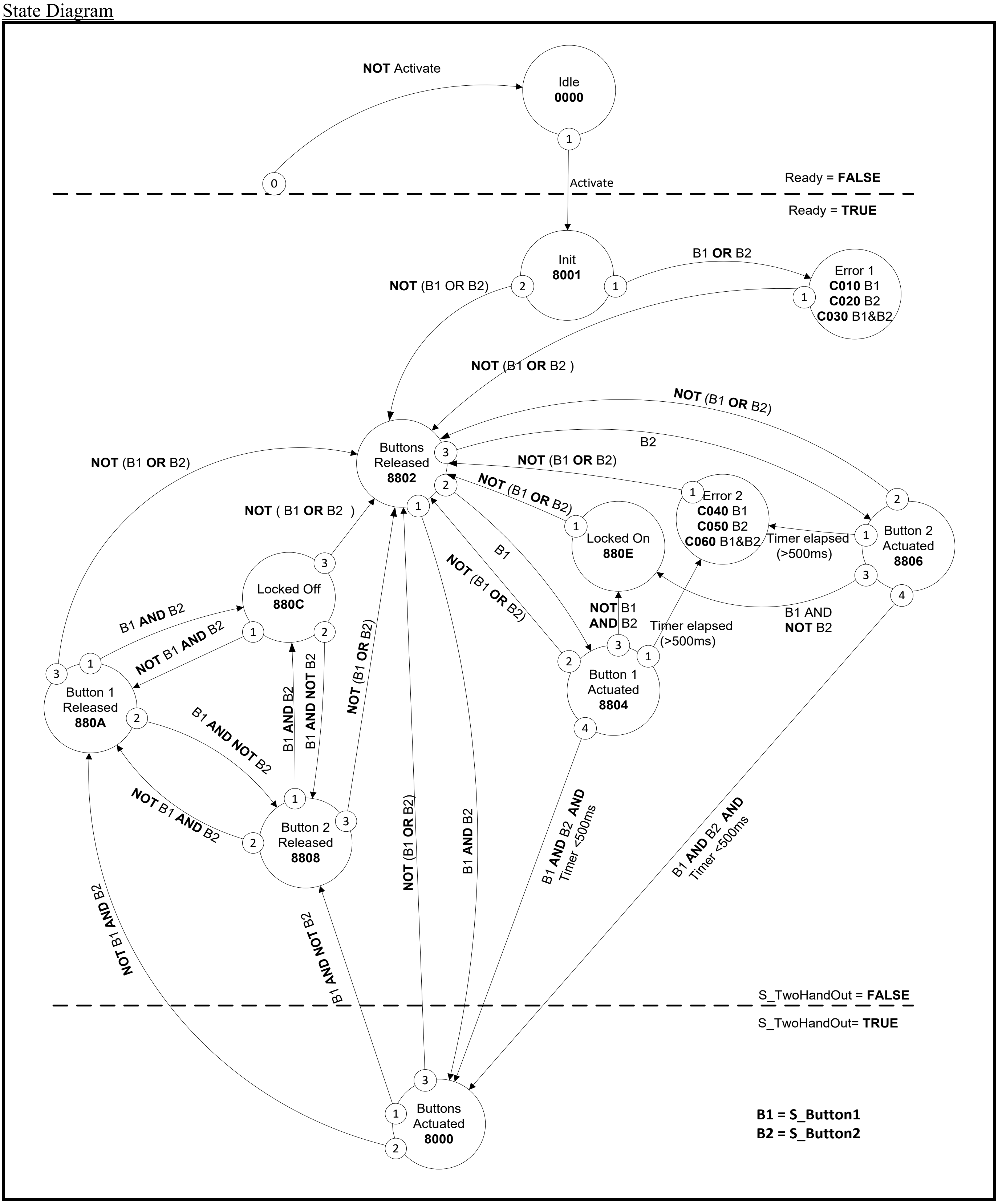
FB Name	**SF_TwoHandControlTypeIII**		
This function block provides the two-hand control functionality for type III A, B and C. The difference is in the input processing of the switches (single contact, 2 switches per input (equivalent), 2 switches per input (antivalent). (see EN 574, Section 4 Type III. Fixed specified time difference is 500 ms.).			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
S_Button1	SAFEBOOL	FALSE	Variable. Input of button 1 (for category 3 or 4: two antivalent contacts) FALSE: Button 1 released. TRUE: Button 1 actuated.
S_Button2	SAFEBOOL	FALSE	Variable. Input of button 2 (for category 3 or 4: two antivalent contacts) FALSE: Button 2 released. TRUE: Button 2 actuated.
VAR_OUTPUT			
Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
S_TwoHandOut	SAFEBOOL	FALSE	Safety related output signal. FALSE: No correct two hand operation. TRUE: S_Button1 and S_Button2 inputs changed from FALSE to TRUE within 500 ms. and no error occurred. The two-hand operation has been performed correctly.
SafetyDemand	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters
Notes: 1. No Reset input or Error output is required, because no test can be performed on both switches. 2. There is a SF_TwoHandControlTypeIIIC defined in Part 4 - Application Specific FBs for Presses, Chapter 4.11 which fulfills category 4 and with additional support for unplugging.			



6.9.3 Functional Description

This function block provides the two-hand control functionality according to EN 574, Section 4 Type III. If S_Button1 and S_Button2 are set to TRUE within 500 ms and in correct sequence, then the S_TwoHandOut output is also set to TRUE. The FB also controls the release of both buttons before setting the output S_TwoHandOut again to TRUE.

State Diagram



Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Figure 38: State diagram for SF_TwoHandControlTypeIII

Typical Timing Diagram

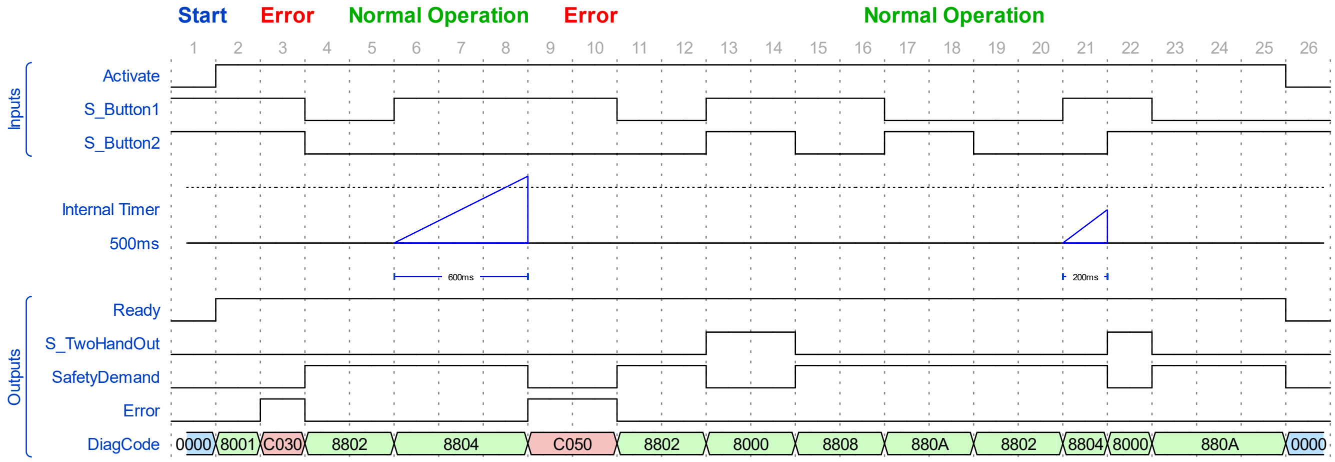


Figure 39: Timing diagram for SF_TwoHandControlTypeIII

6.9.4 Error Detection

After activation of the FB, any button set to TRUE is detected as an invalid input setting leading to an error. The FB detects when the divergence of the input signals exceeds 500 ms.

6.9.5 Error Behavior

In the event of an error, the S_TwoHandOut output is set to FALSE and remains in this safe state.
The Error state is left when both buttons are released (set to FALSE).

6.9.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C010	Error 1 B1	S_Button1 was TRUE on FB activation.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = FALSE
		Error = TRUE

C020	Error 1 B2	S_Button2 was TRUE on FB activation.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = FALSE
		Error = TRUE

C030	Error 1 B1&B2	The signals at S_Button1 and S_Button2 were TRUE on FB activation.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = FALSE
		Error = TRUE

C040	Error 2 B1	S_Button1 was FALSE and S_Button 2 was TRUE after 500 ms in state 8804, 8806.
		Ready = TRUE
		SafetyDemand = FALSE
		Error = TRUE
		S_TwoHandOut = FALSE

C050	Error 2 B2	S_Button1 was TRUE and S_Button 2 was FALSE after 500 ms in state 8804, 8806.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = FALSE
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

C060	Error 2 B1&B2	S_Button1 was TRUE and S_Button 2 was TRUE after 500 ms in state 8804 or 8806. This state is only possible when the states of the inputs (S_Button1 and S_Button2) change from divergent to convergent (both TRUE) simultaneously when the timer elapses (500 ms) at the same cycle.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_TwoHandOut = FALSE
		SafetyDemand = FALSE
		Error = FALSE

8000	Buttons Actuated	Both buttons actuated correctly. The safety related output is enabled.
		Ready = TRUE
		S_TwoHandOut = TRUE
		SafetyDemand = FALSE
		Error = FALSE

8001	Init	Function block is active, but in the Init state.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = FALSE
		Error = FALSE

8802	Buttons Released	No Button is actuated.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

8804	Button 1 Actuated	Only Button 1 is actuated. Start monitoring timer.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

8806	Button 2 Actuated	Only Button 2 is actuated. Start monitoring timer.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

8808	Button 2 Released	The safety related output was enabled and is disabled again.
		FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output.
		In this state, S_Button1 is TRUE and S_Button2 is FALSE after disabling the safety related output.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

880A	Button 1 Released	The safety related output was enabled and is disabled again.
		FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output.
		In this state, S_Button1 is FALSE and S_Button2 is TRUE after disabling the safety related output.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

880C	Locked Off	The safety related output was enabled and is disabled again.
		FALSE at both S_Button1 and S_Button2 was not achieved after disabling the safety related output.
		In this state, S_Button1 is TRUE and S_Button2 is TRUE after disabling the safety related output.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

880E	Locked On	Incorrect actuation of the buttons. Waiting for release of both buttons.
		Ready = TRUE
		S_TwoHandOut = FALSE
		SafetyDemand = TRUE
		Error = FALSE

6.10 Testable Safety Sensor

6.10.1 Applicable Safety Standards

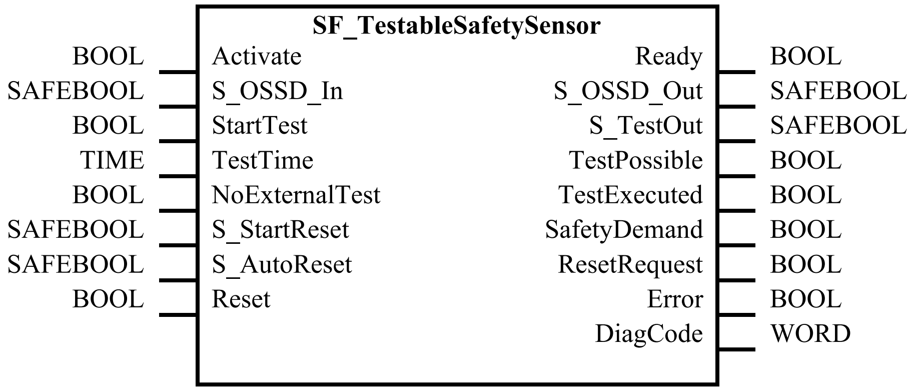
Standards	Requirements
IEC 61496-1: 2012	4.2.2.3 Particular requirements for a type 2 ESPE A type 2 ESPE shall have a means of periodic test to reveal a failure to danger (for example loss of detection capability, response time exceeding that specified). The test shall be performed at power-on of the ESPE before going to the ON-state and at each reset as a minimum. NOTE1: Depending on the application, the periodic test may need to be performed more often to achieve a desired safety performance. A single fault resulting in the loss of detection capability or the increase in response time beyond the specified time or preventing one or more of the OSSDs going to the OFF-state, shall result in a lock-out condition as a result of the next periodic test. Where the periodic test is intended to be initiated by an external (for example machine) safety-related control system, the ESPE shall be provided with suitable input facilities (for example terminals). The duration of the periodic test shall be such that the intended safety function is not impaired. NOTE If the type 2 ESPE is intended for use as a trip device (for example when used as a perimeter guard), and the duration of the periodic test is greater than 150 ms, it is possible for a person to pass through the detection zone without being detected. In this case a restart interlock should be included. If the periodic test is automatically initiated, the correct functioning of the periodic test shall be monitored. In the event of a fault, the OSSD(s) shall be signalled to go to the OFF-state. If one or more OSSDs does not go to the OFF-state, a lock-out condition shall be initiated. An ESPE with only one OSSD shall have a minimum of one SSD (see Clause A.4).
EN ISO 13849-1:2015	5.2.2 Manual reset function
ISO 12100: 2010	6.2.11.4: Restart following power failure/spontaneous restart

Note: the Power-on test of the ESPE is not part of this specification of the FB. The user must deal with this in the application program.

6.10.2 Interface Description

FB Name	**SF_TestableSafetySensor**		
This function block detects, for example, the loss of the sensing unit detection capability, the response time exceeding that specified, and static ON signal in single-channel sensor systems. It can be used for external testable safety sensors (ESPE: Electro-sensitive protective equipment, such as a light beam).			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
S_OSSD_In	SAFEBOOL	FALSE	Variable. Status of sensor output, e.g., light curtain. FALSE: Safety sensor in test state or demand for safety-related response. TRUE: Sensor in the state for normal operating conditions.
StartTest	BOOL	FALSE	Variable. Input to start sensor test. Sets "S_TestOut" and starts the internal time monitoring function in the FB. FALSE: No test requested. TRUE: Test requested.
TestTime	TIME	T#10ms	Constant. Range: 0 … 150ms. Test time of safety sensor.

NoExternalTest	BOOL	FALSE	Constant. Indicates if external manual sensor test is supported. FALSE: The external manual sensor test is supported. Only after a complete manual sensor switching sequence, a automatic test is possible again after a faulty automatic sensor test. TRUE: The external manual sensor test is **not** supported. An automatic test is possible again without a manual sensor switching sequence after faulty automatic sensor test.
S_StartReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
S_AutoReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
Reset	BOOL	FALSE	See Section 5.1.1 General Input Parameters
VAR_OUTPUT			
Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
S_OSSD_Out	SAFEBOOL	FALSE	Safety related output indicating the status of the ESPE. FALSE: The sensor has a safety-related action request or test error. TRUE: The sensor has no safety-related action request AND no test error.
S_TestOut	SAFEBOOL	TRUE	Coupled with the test input of the sensor. Although specified as SAFEBOOL, in practice this signal will often be connected to a BOOL output. FALSE: Test request issued. TRUE: No test request.
TestPossible	BOOL	FALSE	Feedback signal to the process. FALSE: An automatic sensor test is **not** possible. TRUE: An automatic sensor test is possible.
TestExecuted	BOOL	FALSE	A positive signal edge indicates the successful execution of the automatic sensor test. FALSE: - An automatic sensor test was not executed yet. - An automatic sensor test is active. - An automatic sensor test was faulty. TRUE: A sensor test was executed successfully.
SafetyDemand	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
ResetRequest	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters
Notes: OSSD: Output Signal Switching Device.			



6.10.3 Functional Description

Type 2 ESPE shall have a means of periodic testing to detect a hazardous fault (e.g., loss of sensing unit detection capability, response time exceeding that specified). The test signal shall simulate the actuation of the sensing device and the duration of the periodic test shall not exceed 150 ms. The test shall verify that each light beam operates in the manner specified by the supplier. If the periodic test is intended to be initiated by an external safety-related control system (e.g., a machine), the ESPE shall be provided with suitable input facilities (e.g., terminals). The ESPE must be selected in respect of the product standards EN IEC 61496-1, -2 and -3. It must be monitored by separate functionality, that the test is initiated within appropriate intervals.

The S_StartReset and S_AutoReset inputs shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.
Note: the Power-on test of the ESPE is not part of this specification of the FB. The user has to deal with this in the application program.

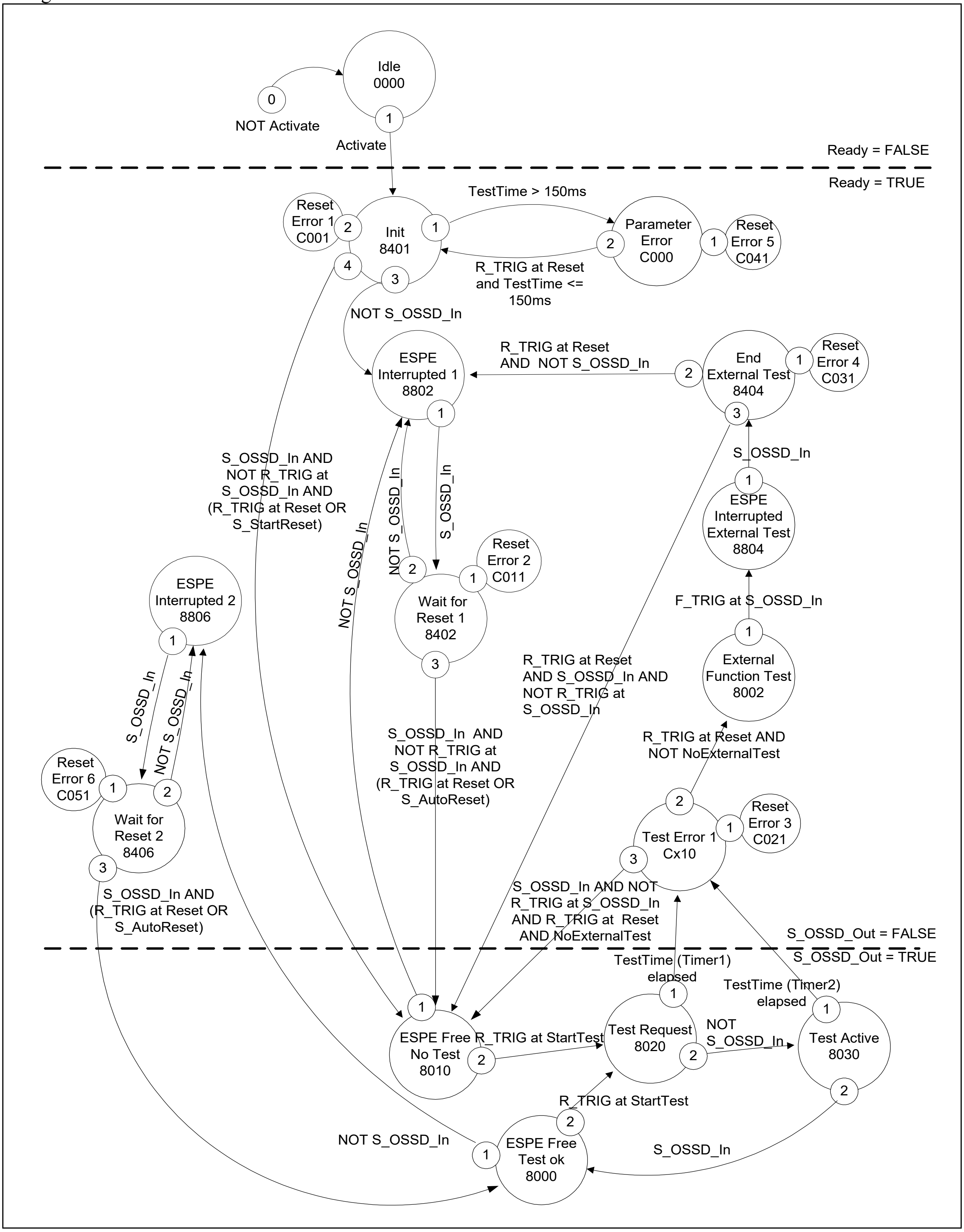
Test mode:

1. StartTest = TRUE: S_TestOut = FALSE. Start monitoring time.
2. S_TestOut signal stops transmitter (Monitoring of TestTime started first time)
3. S_OSSD_In changes from TRUE to FALSE (Monitoring of TestTime started second time)
4. S_TestOut changes from FALSE to TRUE.
5. Start transmitter.
6. Sensor S_OSSD_In changes from FALSE to TRUE
7. Stop monitoring time.
8. S_OSSD_Out is set to TRUE during testing.

Optional startup inhibits:

- Startup inhibit after function block activation.
- Startup inhibit after interruption of the protective device.

State Diagram



Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Figure 40: State diagram for SF_TestableSafetySensor

Typical Timing Diagram

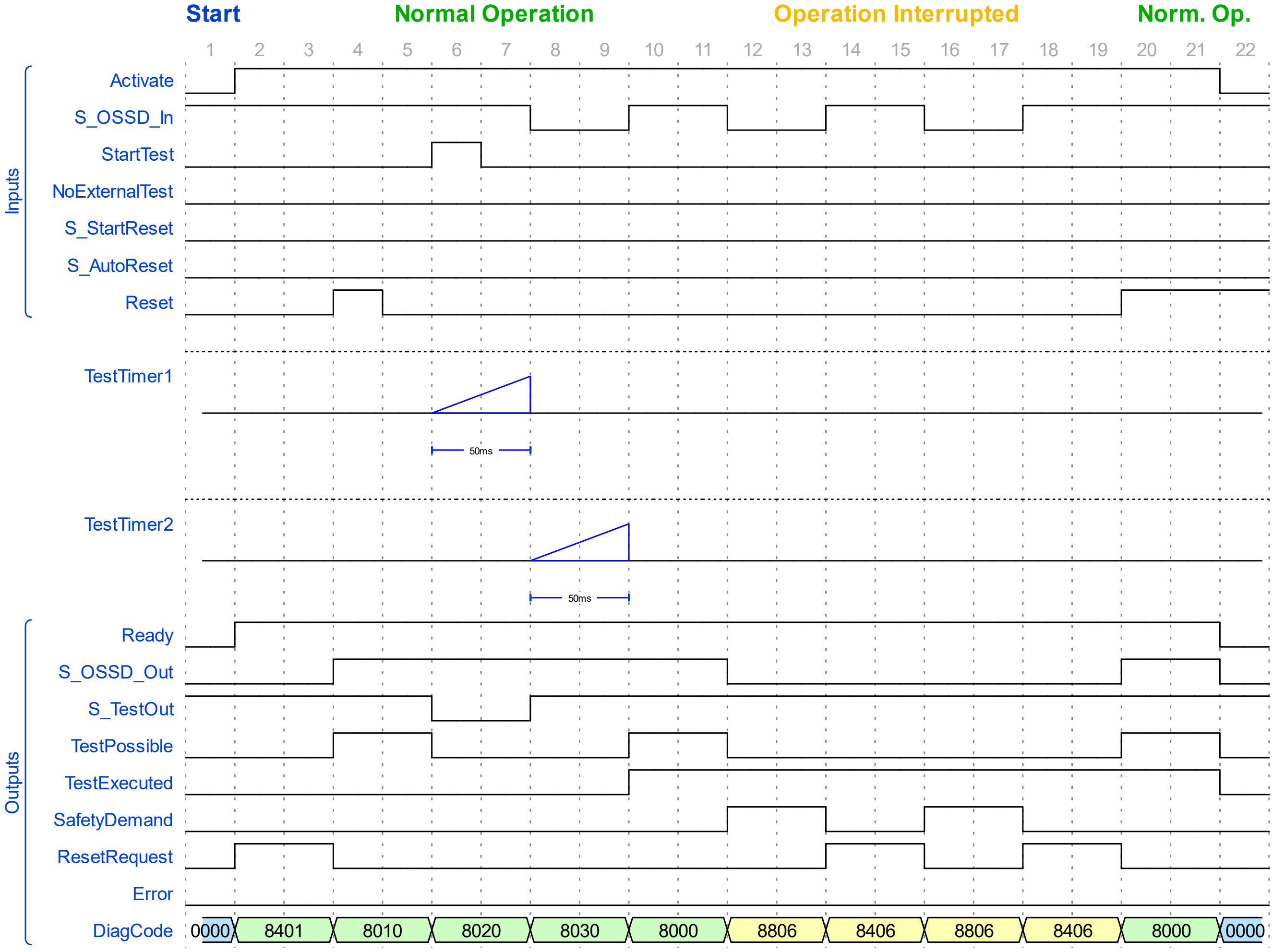


Figure 41: Timing diagram for SF_TestableSafetySensor

6.10.4 Error Detection

The following conditions force a transition to the Error state:

- Test time overrun without delayed sensor feedback.
- Test without sensor signal feedback.
- Invalid static reset signal in the process.
- Plausibility check of the monitoring time setting.

6.10.5 Error Behavior

In the event of an error, the S_OSSD_Out output is set to FALSE and remains in this safe state.
Once the error has been removed and the sensor is on (S_OSSD_In = TRUE) – a reset removes the error state and sets the S_OSSD_Out output to TRUE.
If S_AutoReset = FALSE, a rising trigger is required at Reset.
After transition of S_OSSD_In to TRUE, the optional startup inhibit can be reset by a rising edge at the Reset input.
After block activation, the optional startup inhibit can be reset by a rising edge at the Reset input.

6.10.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C000	Parameter Error	Invalid value at the TestTime parameter.
		Values between 0 ms and 150 ms are possible.
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C001	Reset Error 1	Static Reset condition detected after FB activation.
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C011	Reset Error 2	Static Reset condition detected in state 8402.
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C021	Reset Error 3	Static Reset condition detected in state Cx10.
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C031	Reset Error 4	Static Reset condition detected in state 8404.
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C041	Reset Error 5	Static Reset condition detected in state C000.
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

C051	Reset Error 6	Static Reset condition detected in state 8406.
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

Cx10	Test Error 1	Test time elapsed in state 8020 or 8030.
		IF S_OSSD_IN = TRUE AND NoExternalTest = TRUE THEN x=4 ELSE x=0.
		
		State C410:
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE
		
		State C010:
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8401	Init	An activation has been detected by the FB.
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8802	ESPE Interrupted 1	The FB has detected a safety demand.
		The switch has not been automatically tested yet.
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8402	Wait for Reset 1	Wait for rising trigger of Reset after state 8802.
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

8002	External Function Test	The automatic sensor test was faulty.
		An external manual sensor test is necessary.
		The support for the necessary external manual sensor test has been activated at the FB (NoExternalTest = FALSE).
		A negative signal edge at the sensor is required.
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8804	ESPE Interrupted External Test	The automatic sensor test was faulty.
		An external manual sensor test is necessary.
		The support for the necessary external manual sensor test has been activated at the FB (NoExternalTest = FALSE).
		A TRUE signal at the sensor is required.
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8404	End External Test	The automatic sensor test was faulty.
		An external manual sensor test is necessary.
		The support for the necessary external manual sensor test has been activated at the FB (NoExternalTest = FALSE).
		The external manual test is complete.
		The FB detected a complete sensor switching cycle (external controlled).
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8010	ESPE Free No Test	The FB has not detected a safety demand.
		The sensor has not been tested automatically.
		Ready = TRUE
		S_OSSD_Out = TRUE
		S_TestOut = TRUE
		TestPossible = TRUE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8806	ESPE Interrupted 2	The FB has detected a safety demand.
		The switch was automatically tested.
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = TRUE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8406	Wait for Reset 2	Wait for rising trigger of Reset after state 8806.
		Ready = TRUE
		S_OSSD_Out = FALSE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = TRUE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

8020	Test Request	The automatic sensor test is active. Test Timer is started first time.
		The transmitter signal of the sensor is switched off by the FB.
		The signal of the receiver must follow the signal of the transmitter.
		Ready = TRUE
		S_OSSD_Out = TRUE
		S_TestOut = FALSE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8030	Test Active	The automatic sensor test is active. Test Timer is started second time.
		Timer 1 stopped.
		The transmitter signal of the sensor is switched on by the FB.
		The signal of the receiver must follow the signal of the transmitter.
		Ready = TRUE
		S_OSSD_Out = TRUE
		S_TestOut = TRUE
		TestPossible = FALSE
		TestExecuted = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8000	ESPE Free Test ok	The FB has not detected a safety demand. Timer 2 is stopped.
		The sensor was automatically tested.
		Ready = TRUE
		S_OSSD_Out = TRUE
		S_TestOut = TRUE
		TestPossible = TRUE
		TestExecuted = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

6.11 Sequential Muting

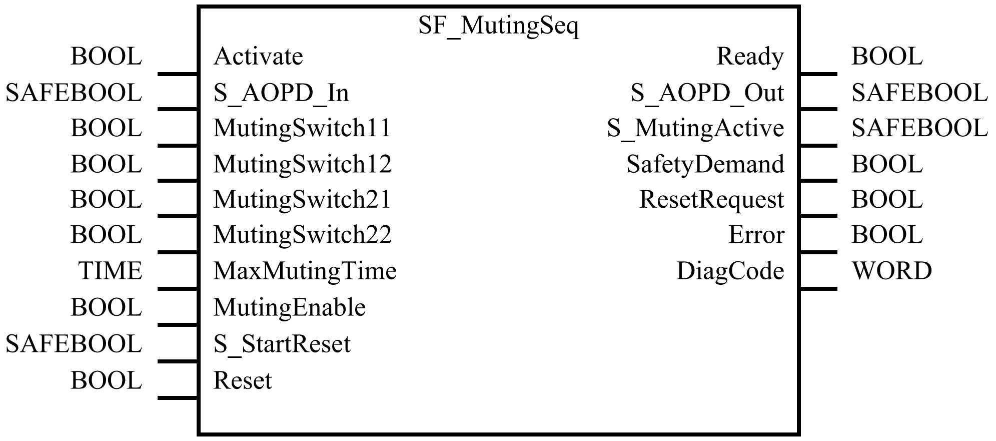
6.11.1 Applicable Safety Standards

Standards	Requirements
IEC 61496-1:2012	A.7 Muting, A.7.1.2 There shall be at least two independent hard-wired muting signal sources to initiate the function. It shall not be possible to initiate muting when the OSSDs are already in the OFF state. A.7.1.3 The mute function shall only be initiated by the correct sequence and/or timing of the mute signals. Should conflicting muting signals occur, the ESPE shall not allow a muted condition to occur. A.7.1.4 There shall be at least two independent hard-wired muting signal sources to stop the function. The muting function shall stop when the first of these muting signals changes state. The deactivation of the muting function shall not rely only on the clearance of the ESPE. A.7.1.5 The muting signals should be continuously present during muting. When the signals are not continuously present, an incorrect sequence and/or the expiration of a pre-set time limit shall cause either a lock-out condition or a restart interlock. A.7.4 Indication: A mute status signal or indicator shall be provided (in some applications, an indication signal of muting is necessary)
IEC / TS 62046 Ed. 2: 2008	5.5. General Application Requirements for Muting
EN ISO 13849-1:2015	5.2.2 Manual reset function
ISO 12100: 2010	6.2.11.4: Restart following power failure/spontaneous restart

6.11.2 Interface Description

FB Name	**SF_MutingSeq**			
Muting is the intended suppression of the safety function (e.g., light barriers). In this FB, sequential muting with four muting sensors is specified.				
VAR_INPUT				
	Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
	Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
	S_AOPD_In	SAFEBOOL	FALSE	Variable. OSSD signal from AOPD. FALSE: Protection field interrupted. TRUE: Protection field not interrupted.
	MutingSwitch11	BOOL	FALSE	Variable. Status of Muting sensor 11. FALSE: Muting sensor 11 not actuated. TRUE: Workpiece actuates muting sensor 11. Depending on the safety requirements it can be necessary to connect a SAFEBOOL.
	MutingSwitch12	BOOL	FALSE	Variable. Status of Muting sensor 12. FALSE: Muting sensor 12 not actuated. TRUE: Workpiece actuates muting sensor 12. Depending on the safety requirements it can be necessary to connect a SAFEBOOL.
	MutingSwitch21	BOOL	FALSE	Variable. Status of Muting sensor 21. FALSE: Muting sensor 21 not actuated. TRUE: Workpiece actuates muting sensor 21. Depending on the safety requirements it can be necessary to connect a SAFEBOOL.

MutingSwitch22	BOOL	FALSE	Variable. Status of Muting sensor 22. FALSE: Muting sensor 22 not actuated. TRUE: Workpiece actuates muting sensor 22. Depending on the safety requirements it can be necessary to connect a SAFEBOOL.
MaxMutingTime	TIME	T#0s	Constant 0 .. 120 min (application specific); Maximum time for complete muting sequence, timer started when second muting sensor is actuated. If needed this can be combined with SF_Override.
MutingEnable	BOOL	FALSE	Variable or constant. Command by the control system that enables the start of the muting function when needed by the machine cycle. After the start of the muting function, this signal can be switched off. FALSE: Muting not enabled TRUE: Start of Muting function enabled
S_StartReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
Reset	BOOL	FALSE	See Section 5.1.1 General Input Parameters
VAR_OUTPUT			
Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
S_AOPD_Out	SAFEBOOL	FALSE	Safety related output. It indicates the status of the muted guard. FALSE: AOPD protection field interrupted and muting not active. TRUE: AOPD protection field not interrupted or muting active.
S_MutingActive	SAFEBOOL	FALSE	Indicates the status of Muting process. FALSE: Muting not active. TRUE: Muting active.
SafetyDemand	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
ResetRequest	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters
Notes: A short circuit in the muting sensor signals, or a functional application error to supply these signals, are not detected by this FB but interpreted as incorrect muting sequence (The types are BOOL, provided by the functional application hardware and / or software). However, this condition should not lead to unwanted muting. The user should take care to include this in his risk analysis.			



6.11.3 Functional Description

Muting is the intended suppression of the safety function. This is required, e.g., when transporting the material into the danger zone without causing the machine to stop. Muting is triggered by muting sensors. The use of two or four muting sensors and correct integration into the production sequence must ensure that no persons enter the danger zone while the light curtain is muted. Muting sensors can be proximity switches, photoelectric barriers, limit switches, etc. which do not have to be failsafe. Active muting mode must be indicated by indicator lights.

There are sequential and parallel muting procedures. In this FB, sequential muting with four muting sensors was used; an explanation for the forward direction of transportation is provided below. The FB can be used in both directions, forward and backward. The muting should be enabled with the MutingEnable signal by the process control to avoid manipulation. When the MutingEnable signal is not available, this input must be set to TRUE.

The FB input parameters include the signals of the four muting sensors (MutingSwitch11 ... MutingSwitch22) as well as the OSSD signal from the "active opto-electronic protective device", S_AOPD_In.

The S_StartReset input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

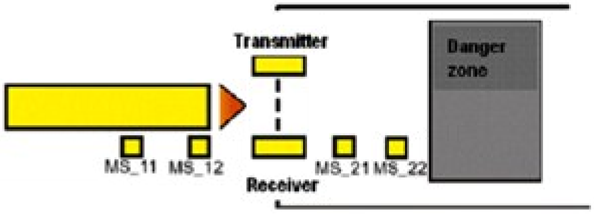
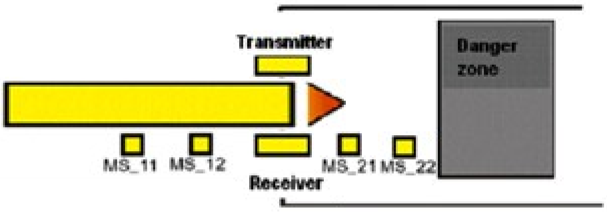
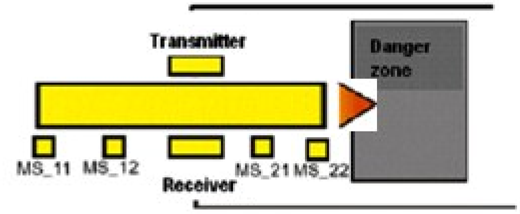
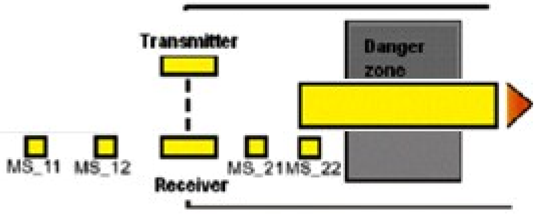
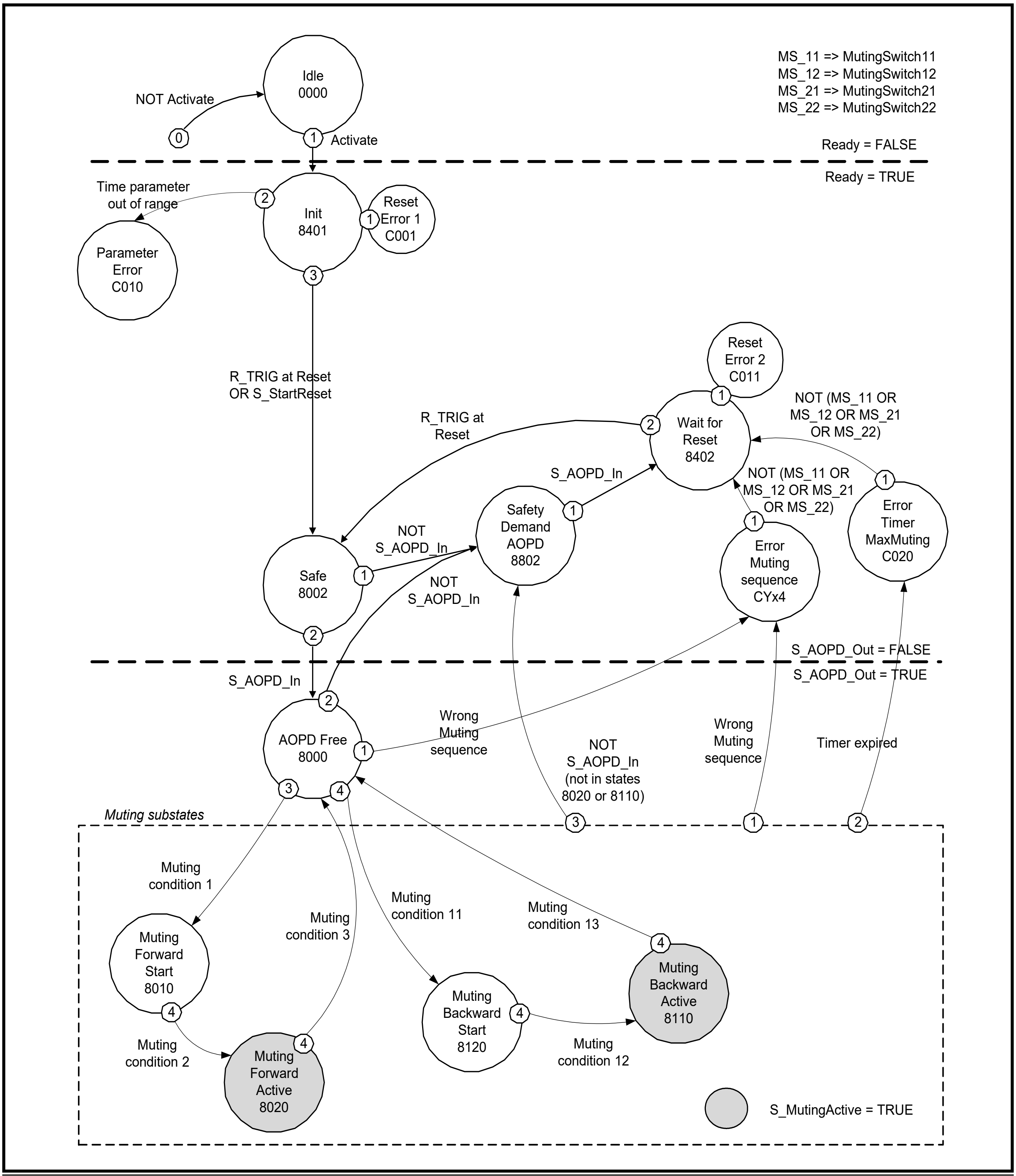
No.	Figure	Explanation
1	 Transmitter Danger zone MS_11 MS_12 MS_21 MS_22 Receiver	If muting sensor MutingSwitch12 (MS_12) is activated by the product after MutingSwitch11 (MS_11), the muting mode is activated and the MaxMutingTime timer is started.
2	 Transmitter Danger zone MS_11 MS_12 MS_21 MS_22 Receiver	Muting mode remains active as long as MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are activated by the product. The product may pass through the light curtain without causing a machine stop.
3	 Transmitter Danger zone MS_11 MS_12 MS_21 MS_22 Receiver	Before muting sensors MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are disabled, muting sensors MutingSwitch21 (MS_21) and MutingSwitch22 (MS_22) must be activated. This ensures that muting mode remains active.
4	 Transmitter Danger zone MS_11 MS_12 MS_21 MS_22 Receiver	Muting mode is terminated if only muting sensor MutingSwitch22 (MS_22) is activated by the product.

Figure 42: Example for SF_MutingSeq in forward direction with four sensors

State Diagram



Note 1: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Note 2: Within muting substates, transitions due to Error Muting sequence (priority 1), Error Timer (priority 2), Safety demand AOPD (priority 3) have higher priority than transitions to Muting substates (priority 4).

Figure 43: State diagram for SF_MutingSeq

Muting Conditions

Forward Direction

Muting condition 1 (to 8010) (MS_11 is the first entry switch actuated)**:**
MutingEnable AND (**R_TRIG at** MS_11 **AND NOT** MS_12 **AND NOT** MS_21 **AND NOT** MS_22)

Muting condition 2 (from 8010 to 8020) (MS_12 is the second entry switch actuated). Start timer MaxMutingTime**:**
MutingEnable AND (MS_11 **AND R_TRIG at** MS_12 **AND NOT** MS_21 **AND NOT** MS_22)

Muting condition 3 (from 8020 to 8000) (MS_21 is the first exit switch released). Stop timer MaxMutingTime**:**
NOT MS_11 **AND NOT** MS_12 **AND F_TRIG at** MS_21 **AND** MS_22

Backward Direction

Muting condition 11 (to 8120) (MS_22 is the first entry switch actuated)**:**
MutingEnable AND (**NOT** MS_11 **AND NOT** MS_12 **AND NOT** MS_21 **AND R_TRIG at** MS_22)

Muting condition 12 (from 8120 to 8110) (MS_21 is the second entry switch actuated). Start timer MaxMutingTime:
MutingEnable AND (**NOT** MS_11 **AND NOT** MS_12 **AND R_TRIG at** MS_21 **AND** MS_22)

Muting condition 13 (MS_12 is the first exit switch released). Stop timer MaxMutingTime**:**
MS_11 **AND F_TRIG at** MS_12 **AND NOT** MS_21 **AND NOT** MS_22

Specification of wrong Muting Sequences:

In 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)

In state 8010: **NOT** MutingEnable **OR NOT** MS_11 **OR** MS_21 **OR** MS_22

In state 8020: **R_TRIG at** MS_11 **OR R_TRIG at** MS_12 **OR F_TRIG at** MS_22
OR (MS_11 **AND F_TRIG at** MS_12)
OR ((MS_11 **OR** MS_12) **AND** (**F_TRIG** at MS_21))
OR ((**NOT** MS_11 **OR NOT** MS_12) **AND NOT** MS_22)
OR ((**NOT** MS_11 **OR NOT** MS_12 **OR NOT** MS_21) **AND R_TRIG at** MS_22)
OR ((MS_11 **AND** MS_22) **AND** (**NOT** MS_12 **OR NOT** MS_21))
OR (**R_TRIG at** MS_21 **AND R_TRIG at** MS_22)
OR (**F_TRIG at** MS_11 **AND F_TRIG at** MS_12)
OR (**F_TRIG at** MS_12 **AND F_TRIG at** MS_21)
OR (**NOT** MS_11 **AND** MS_12 **AND NOT** MS_21)
In state 8120: **NOT** MutingEnable **OR** MS_11 **OR** MS_12 **OR NOT** MS_22

In state 8110: **F_TRIG at** MS_11 **OR R_TRIG at** MS_21 **OR R_TRIG at** MS_22
OR (MS_22 **AND F_TRIG** at MS_21)
OR ((MS_22 **OR** MS_21) **AND** (**F_TRIG** at MS_12))
OR ((**NOT** MS_22 **OR NOT** MS_21) **AND NOT** MS_11)
OR ((**NOT** MS_22 **OR NOT** MS_21 **OR NOT** MS_12) **AND R_TRIG at** MS_11)
OR ((MS_11 **AND** MS_22) **AND** (**NOT** MS_12 **OR NOT** MS_21))
OR (**R_TRIG at** MS_11 **AND R_TRIG at** MS_12)
OR (**F_TRIG at** MS_22 **AND F_TRIG at** MS_21)
OR (**F_TRIG** at MS_21 **AND F_TRIG** at MS_12)
OR (**NOT** MS_12 **AND** MS_21 **AND NOT** MS_22)

Typical Timing Diagram

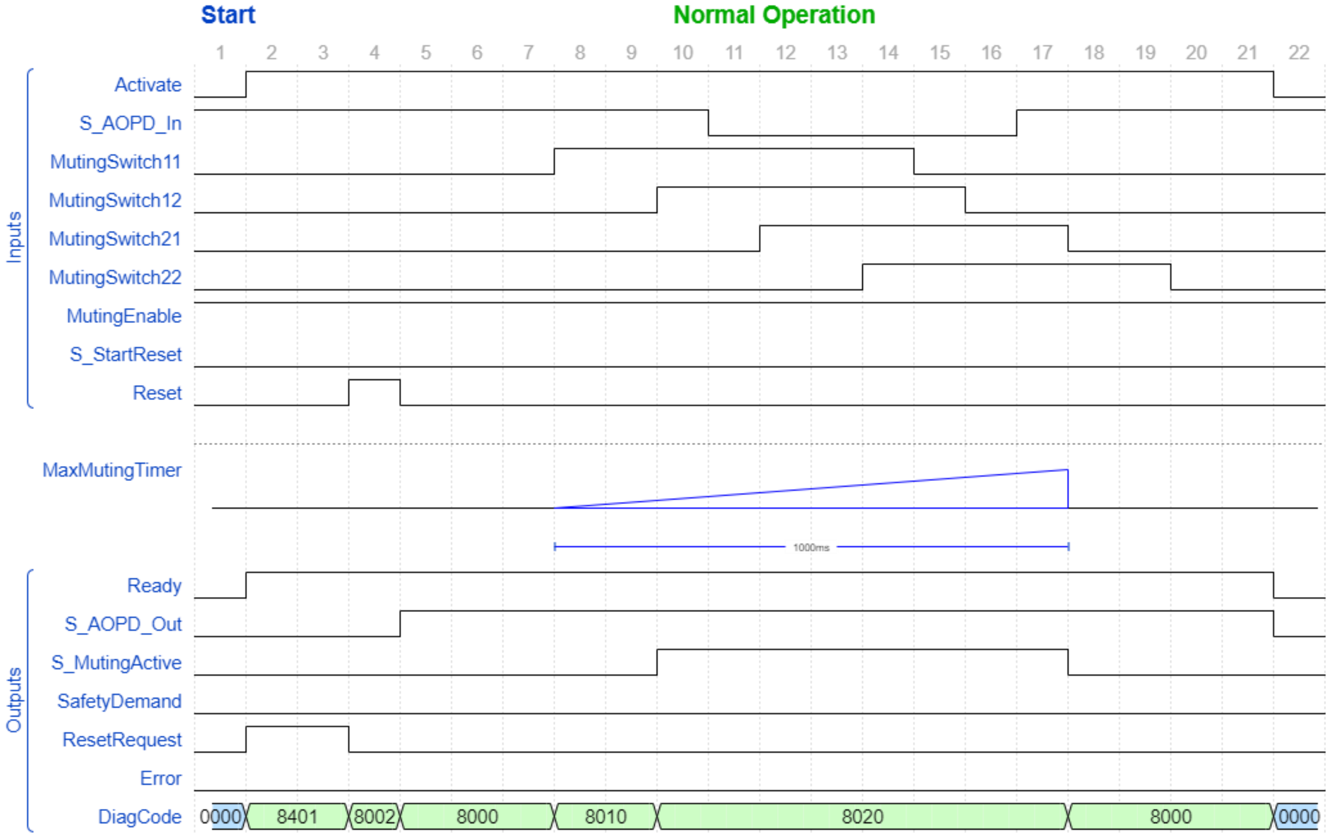


Figure 44: Timing diagram for SF_MutingSeq with manual reset

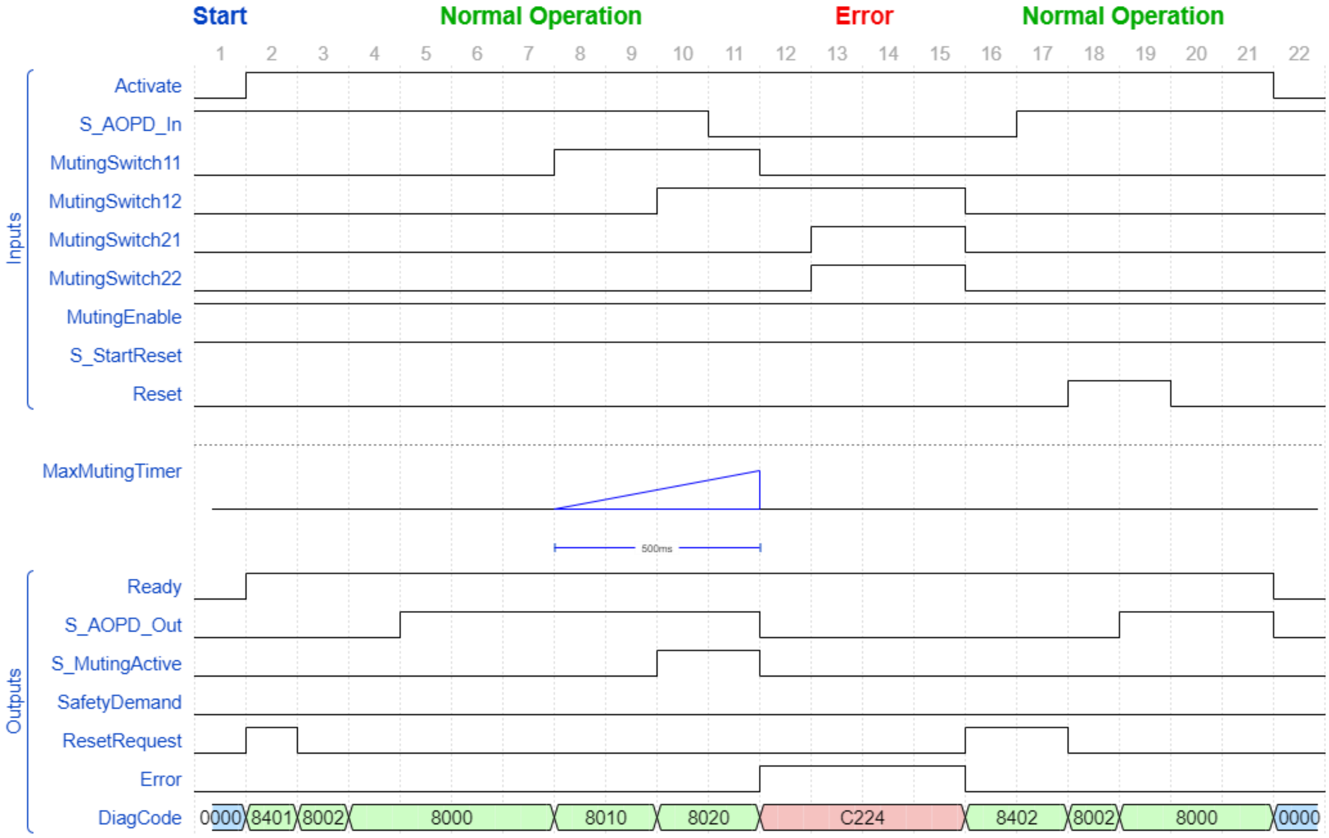


Figure 45: Timing diagram for SF_MutingSeq with S_StartReset = TRUE

6.11.4 Error Detection

The FB detects the following error conditions:

- Muting sensors MutingSwitch11, MutingSwitch12, MutingSwitch21, and MutingSwitch22 are activated in the wrong order.
- Muting sequence starts without being enabled by MutingEnable.
- A static Reset condition.
- MaxMutingTime has been set to a value less than T#0s or greater than T#120min.
- The muting function (S_MutingActive = TRUE) exceeds the maximum muting time MaxMutingTime.

6.11.5 Error Behavior

In the event of an error, the S_AOPD_Out and S_MutingActive outputs are set to FALSE. The DiagCode output indicates the relevant error code, and the Error output is set to TRUE.

A restart is inhibited until the error conditions are cleared and the Safe state is acknowledged with Reset by the operator.

6.11.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C001	Reset Error 1	Static Reset condition detected after FB activation.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C011	Reset Error 2	Static Reset condition detected in state 8402.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

CYx4	Error Muting sequence	Error detected in muting sequence in states 8000, 8010, 8020, 8120 or 8110. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE Y = Status in the sequence (2 states for forward and 2 states for backward direction). C0x4 = Error occurred in state 8000 C1x4 = Error occurred in state Forward 8010 C2x4 = Error occurred in state Forward 8020 C3x4 = Error occurred in state Backward 8120 C4x4 = Error occurred in state Backward 8110 CFx4 = Muting Enable missing x = Status of the sensors when error occurred (4 bits: LSB = MS_11; MS_12; MS_21; MSB = MS_22).

DiagCode	State Name	State Description and Output Setting

C010	Parameter Error	MaxMutingTime value out of range.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C020	Error Timer MaxMuting	Timing error: Active muting time (when S_MutingActive = TRUE) exceeds MaxMutingTime.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8000	AOPD Free	Muting not active and no safety demand from AOPD.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8401	Init	Function block has been activated.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

8802	Safety Demand AOPD	Safety demand detected by AOPD, muting not active.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8402	Wait for Reset	Safety demand or errors have been detected and are now cleared. Operator acknowledgment by Reset required.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8002	Safe	Safety function activated.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8010	Muting Forward Start	Muting forward, sequence is in starting phase and no safety demand.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8020	Muting Forward Active	Muting forward, sequence is active.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8110	Muting Backward Active	Muting backward, sequence is active.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8120	Muting Backward Start	Muting backward, sequence is in starting phase and no safety demand.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

6.12 Parallel Muting

6.12.1 Applicable Safety Standards

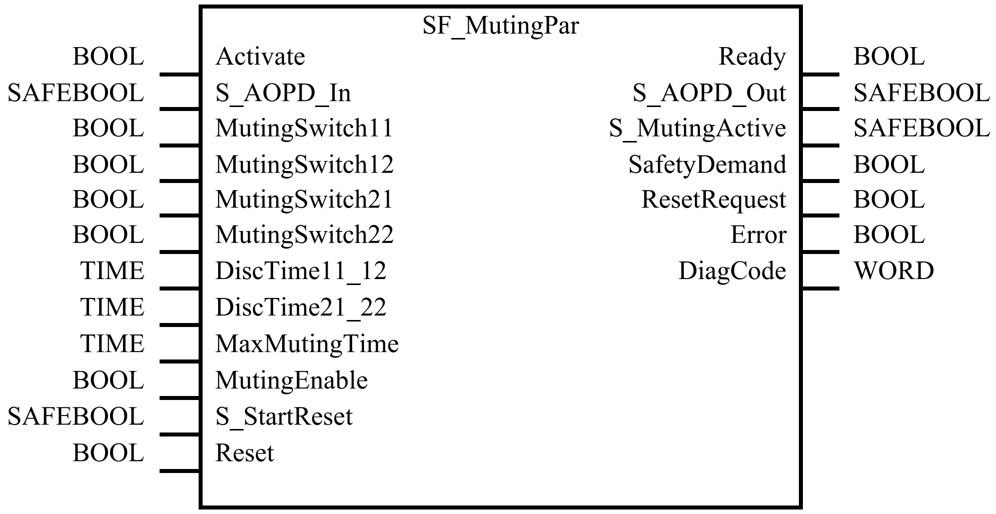
Standards	**Requirements**
IEC 61496-1:2012	A.7 Muting, A.7.1.2 There shall be at least two independent hard-wired muting signal sources to initiate the function. It shall not be possible to initiate muting when the OSSDs are already in the OFF state. A.7.1.3 The mute function shall only be initiated by the correct sequence and/or timing of the mute signals. Should conflicting muting signals occur, the ESPE shall not allow a muted condition to occur. A.7.1.4 There shall be at least two independent hard-wired muting signal sources to stop the function. The muting function shall stop when the first of these muting signals changes state. The deactivation of the muting function shall not rely only on the clearance of the ESPE. A.7.1.5 The muting signals should be continuously present during muting. When the signals are not continuously present, an incorrect sequence and/or the expiration of a pre-set time limit shall cause either a lock-out condition or a restart interlock. A.7.4 Indication: A mute status signal or indicator shall be provided (in some applications, an indication signalof muting is necessary
IEC / TS 62046/Ed. 2: 2008	5.5. General Application Requirements for Muting
EN ISO 13849-1:2015	5.2.2 Manual reset function
ISO 12100: 2010	6.2.11.4: Restart following power failure/spontaneous restart

6.12.2 Interface Description

FB Name	**SF_MutingPar**
Muting is the intended suppression of the safety function. In this FB, parallel muting with four muting sensors is specified.	
VAR_INPUT	

Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
S_AOPD_In	SAFEBOOL	FALSE	Variable. OSSD signal from AOPD. FALSE: Protection field interrupted. TRUE: Protection field not interrupted.
MutingSwitch11	BOOL	FALSE	Variable. Status of Muting sensor 11. FALSE: Muting sensor 11 not actuated. TRUE: Workpiece actuates muting sensor 11. Depending on the safety requirements it can be necessary to connect a SAFEBOOL.
MutingSwitch12	BOOL	FALSE	Variable. Status of Muting sensor 12. FALSE: Muting sensor 12 not actuated. TRUE: Workpiece actuates muting sensor 12. Depending on the safety requirements it can be necessary to connect a SAFEBOOL.
MutingSwitch21	BOOL	FALSE	Variable. Status of Muting sensor 21. FALSE: Muting sensor 21 not actuated. TRUE: Workpiece actuates muting sensor 21. Depending on the safety requirements it can be necessary to connect a SAFEBOOL.

	MutingSwitch22	BOOL	FALSE	Variable. Status of Muting sensor 22. FALSE: Muting sensor 22 not actuated. TRUE: Workpiece actuates muting sensor 22. Depending on the safety requirements it can be necessary to connect a SAFEBOOL.
	DiscTime11_12	TIME	T#0s	Constant 0..4 s; Maximum discrepancy time for MutingSwitch11 and MutingSwitch12.
	DiscTime21_22	TIME	T#0s	Constant 0..4 s; Maximum discrepancy time for MutingSwitch21 and MutingSwitch22.
	MaxMutingTime	TIME	T#0s	Constant 0..120 min (application area specific); Maximum time for complete muting sequence, timer started when first 2 muting sensors are actuated.
	MutingEnable	BOOL	FALSE	Variable or constant. Command by the control system that enables the start of the muting function when needed by the machine cycle. After the start of the muting function, this signal can be switched off. FALSE: Muting not enabled TRUE: Start of Muting function enabled
	S_StartReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
	Reset	BOOL	FALSE	See Section 5.1.1 General Input Parameters
VAR_OUTPUT				
	Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
	S_AOPD_Out	SAFEBOOL	FALSE	Safety related output indicates the status of the muted guard. FALSE: AOPD protection field interrupted and muting not active. TRUE: AOPD protection field not interrupted or muting active.
	S_MutingActive	SAFEBOOL	FALSE	Indicates status of Muting process. FALSE: Muting not active. TRUE: Muting active.
	SafetyDemand	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
	ResetRequest	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
	Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
	DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters
Notes: A short circuit in the muting sensor signals, or a functional application error to supply these signals, are not detected by this FB (The types are BOOL, provided by the functional application hardware and / or software). However, this condition should not lead to unwanted muting. The user should take care to include this in his risk analysis.				



6.12.3 Functional Description

Muting is the intended suppression of the safety function. This is required, e.g., when transporting the material into the danger zone without causing the machine to stop. Muting is triggered by muting sensors. The use of two or four muting sensors and correct integration into the production sequence must ensure that no persons enter the danger zone while the light curtain is muted. Muting sensors can be proximity switches, photoelectric barriers, limit switches, etc. which do not have to be failsafe. Active muting mode must be indicated by indicator lights.

There are sequential and parallel muting procedures. In this FB, parallel muting with four muting sensors is used; an explanation is provided below. The FB can be used in both directions, forward and backward. The muting should be enabled with the MutingEnable signal by the process control to avoid manipulation.

The FB input parameters include the signals of the four muting sensors (MutingSwitch11 ... MutingSwitch22), the OSSD signal from the "active opto-electronic protective device", S_AOPD_In, as well as three parameterizable times (DiscTime11_12, DiscTime21_22, and MaxMutingTime).

The S_StartReset input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

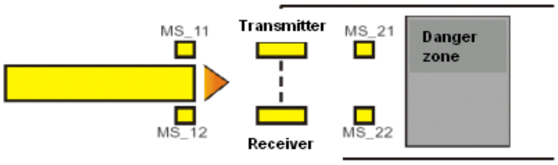
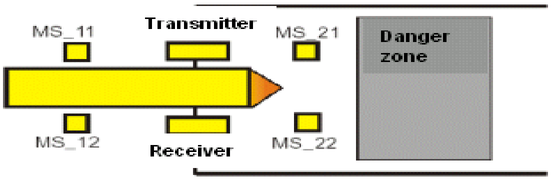
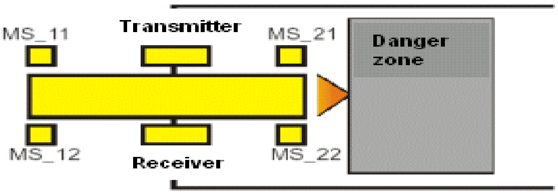
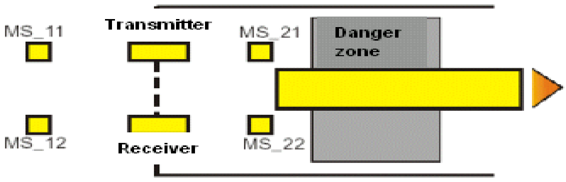
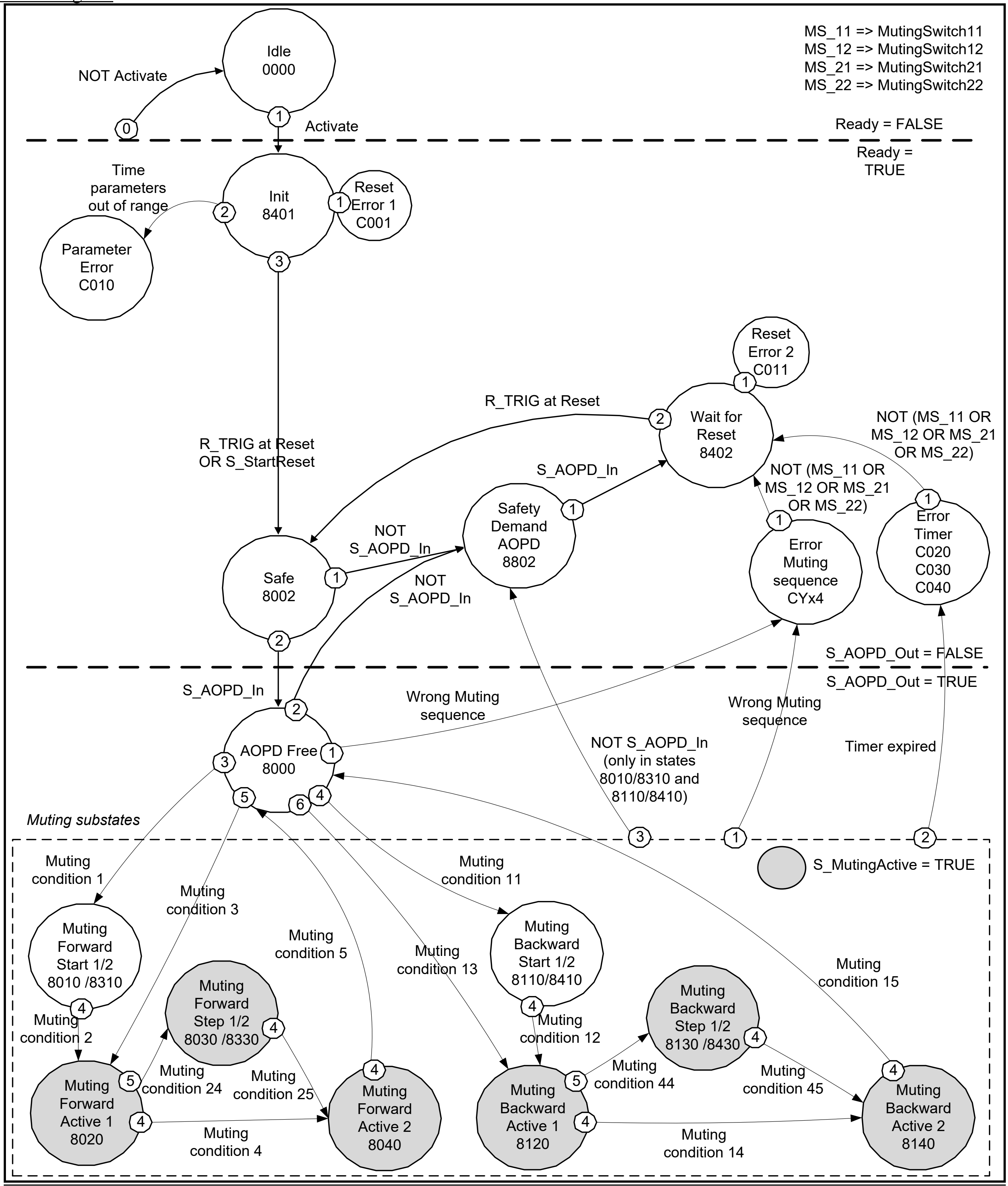
No.	Figure	Explanation
1	 MS_11 Transmitter MS_21 Danger zone MS_12 Receiver MS_22	If the muting sensors MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are activated by the product within the time DiscTime11_12, muting mode is activated (S_MutingActive = TRUE). The MaxMutingTime starts with the first actuated muting sensor.
2	 MS_11 Transmitter MS_21 Danger zone MS_12 Receiver MS_22	Muting mode remains active as long as MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are activated by the product. The product may pass through the light curtain without causing a machine stop.
3	 MS_11 Transmitter MS_21 Danger zone MS_12 Receiver MS_22	Before muting sensors MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are disabled, muting sensors MutingSwitch21 (MS_21) and MutingSwitch22 (MS_22) must be activated. This ensures that muting mode remains active. The time discrepancy between switching of MutingSwitch21 and MutingSwitch22 is monitored by the time DiscTime21_22.
4	 MS_11 Transmitter MS_21 Danger zone MS_12 Receiver MS_22	Muting mode is terminated if either muting sensor MutingSwitch21 (MS_21) or MutingSwitch22 (MS_22) is disabled by the product. The maximum time for muting mode to be active is the MaxMutingTime.

Figure 46: Example for SF_MutingPar in forward direction with four sensors

State Diagram



Note1: The transition from any state to the Idle state due to Activate = FALSE is not shown. However these transitions have the highest priority (0).

Note 2: Within muting substates, transitions due to Error Muting sequence (priority 1), Error Timer (priority 2), Safety demand AOPD (priority 3) have higher priority than transitions to Muting substates (priority 4 or 5).

Note 3: Muting conditions are defined below.

Figure 47: State diagram for SF_MutingPar

Forward Direction

Muting condition 1 (to 8010) (MS_11 is the first entry switch actuated). Start timers MaxMutingTime and DiscTime11_12**:** MutingEnable **AND** (**R_TRIG at** MS_11 **AND NOT** MS_12 **AND NOT** MS_21 **AND NOT** MS_22)
Muting condition 1 (to 8310) (MS_12 is the first entry switch actuated). Start timers MaxMutingTime and DiscTime11_12**:** MutingEnable **AND** (**NOT** MS_11 **AND R_TRIG at** MS_12 **AND NOT** MS_21 **AND NOT** MS_22)

Muting condition 2 (from 8010) (MS_12 is the second entry switch actuated). Stop timer DiscTime11_12**:** MutingEnable **AND** (MS_11 **AND R_TRIG at** MS_12 **AND NOT** MS_21 **AND NOT** MS_22)
Muting condition 2 (from 8310) (MS_11 is the second entry switch actuated). Stop timer DiscTime11_12**:** MutingEnable **AND** (**R_TRIG at** MS_11 **AND** MS_12 **AND NOT** MS_21 **AND NOT** MS_22)

Muting condition 3 (both entry switches actuated in same cycle). Start timer MaxMutingTime**:** MutingEnable **AND** (**R_TRIG at** MS_11 **AND R_TRIG at** MS_12 **AND NOT** MS_21 **AND NOT** MS_22)

Muting condition 4 (all switches actuated)**:** MS_11 **AND** MS_12 **AND** MS_21 **AND** MS_22
Muting condition 24 (to 8030) (MS_21 is the first exit switch actuated). Start timer DiscTime21_22**:** MS_11 **AND** MS_12 **AND R_TRIG at** MS_21 **AND NOT** MS_22
Muting condition 24 (to 8330) (MS_22 is the first exit switch actuated). Start timer DiscTime21_22**:** MS_11 **AND** MS_12 **AND NOT** MS_21 **AND R_TRIG at** MS_22
Muting condition 25 (from 8030) (MS_22 is the second exit switch actuated). Stop timer DiscTime21_22**:** MS_11 **AND** MS_12 **AND** MS_21 **AND R_TRIG at** MS_22
Muting condition 25 (from 8330) (MS_21 is the second exit switch actuated). Stop timer DiscTime21_22**:** MS_11 **AND** MS_12 **AND R_TRIG at** MS_21 **AND** MS_22

Muting condition 5 (one of the exit switches released). Stop timer MaxMutingTime**: NOT** MS_11 **AND NOT** MS_12 **AND (F_TRIG at** MS_21 **OR F_TRIG at** MS_22)

Backward Direction

Muting condition 11 (to 8110) (MS_21 is the first entry switch actuated). Start timers MaxMutingTime and DiscTime21_22**:** MutingEnable AND (**NOT** MS_22 **AND R_TRIG at** MS_21 **AND NOT** MS_11 **AND NOT** MS_12)
Muting condition 11 (to 8410) (MS_22 is the first entry switch actuated). Start timers MaxMutingTime and DiscTime21_22**:** MutingEnable AND (**R_TRIG at** MS_22 **AND NOT** MS_21 **AND NOT** MS_11 **AND NOT** MS_12)

Muting condition 12 (from 8110) (MS_22 is the second entry switch actuated). Stop timer DiscTime21_22: MutingEnable AND (MS_21 **AND R_TRIG at** MS_22 **AND NOT** MS_11 **AND NOT** MS_12)
Muting condition 12 (from 8410) (MS_21 is the second entry switch actuated) . Stop timer DiscTime21_22: MutingEnable AND (**R_TRIG at** MS_21 **AND** MS_22 **AND NOT** MS_11 **AND NOT** MS_12)

Muting condition 13 (both entry switches actuated in same cycle). Start timer MaxMutingTime**:** MutingEnable AND (**R_TRIG at** MS_21 **AND R_TRIG at** MS_22 **AND NOT** MS_11 **AND NOT** MS_12)

Muting condition 14 (all switches actuated)**:** MS_11 **AND** MS_12 **AND** MS_21 **AND** MS_22
Muting condition 44 (to 8130) (MS_11 is the first exit switch actuated). Start timer DiscTime11_12**:** MS_21 **AND** MS_22 **AND R_TRIG at** MS_11 **AND NOT** MS_12
Muting condition 44 (to 8430) (MS_12 is the first exit switch actuated). Start timer DiscTime11_12**:** MS_21 **AND** MS_22 **AND NOT** MS_11 **AND R_TRIG at** MS_12
Muting condition 45 (from 8130) (MS_12 is the second exit switch actuated). Stop timer DiscTime11_12**:** MS_21 **AND** MS_22 **AND** MS_11 **AND R_TRIG at** MS_12
Muting condition 45 (from 8430) (MS_11 is the second exit switch actuated). Stop timer DiscTime11_12**:** MS_21 **AND** MS_22 **AND R_TRIG at** MS_11 **AND** MS_12

Muting condition 15 (one of the exit switches released). Stop timer MaxMutingTime**: NOT** MS_21 **AND NOT** MS_22 **AND (F_TRIG at** MS_11 **OR F_TRIG at** MS_12)

Wrong Muting Sequences:

State 8000:

(MutingEnable = FALSE when muting sequence starts) 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 8010: NOT MutingEnable OR **NOT** MS_11 **OR** MS_21 **OR** MS_22

State 8310: NOT MutingEnable OR **NOT** MS_12 **OR** MS_21 **OR** MS_22

State 8020: **NOT** MS_11 **OR** NOT MS_12

State 8040: **R_TRIG at** MS_11 **OR R_TRIG at** MS_12 **OR R_TRIG at** MS_21 **OR R_TRIG at** MS_22

OR ((MS_11 **OR** MS_12) **AND** (**F_TRIG at** MS_21 **OR F_TRIG at** MS_22))

OR ((**F_TRIG at** MS_11 **OR F_TRIG at** MS_12) **AND** (**F_TRIG at** MS_21 **OR F_TRIG at** MS_22))

State 8030: **NOT** MS_11 **OR NOT** MS_12 **OR NOT** MS_21

State 8330: **NOT** MS_11 **OR NOT** MS_12 **OR NOT** MS_22

State 8110: NOT MutingEnable OR MS_11 **OR** MS_12 **OR NOT** MS_21

State 8410: NOT MutingEnable OR MS_11 **OR** MS_12 **OR NOT** MS_22

State 8120: **NOT** MS_21 **OR NOT** MS_22

State 8140: **R_TRIG at** MS_11 **OR R_TRIG at** MS_12 **OR R_TRIG at** MS_21 **OR R_TRIG at** MS_22

OR ((MS_21 OR MS_22) **AND** (**F_TRIG at** MS_11 **OR F_TRIG at** MS_12))

OR ((**F_TRIG at** MS_11 **OR F_TRIG at** MS_12) **AND** (**F_TRIG at** MS_21 **OR F_TRIG at** MS_22))

State 8130: **NOT** MS_21 **OR NOT** MS_22 **OR NOT** MS_11

State 8430: **NOT** MS_21 **OR NOT** MS_22 **OR NOT** MS_12

Typical Timing Diagram

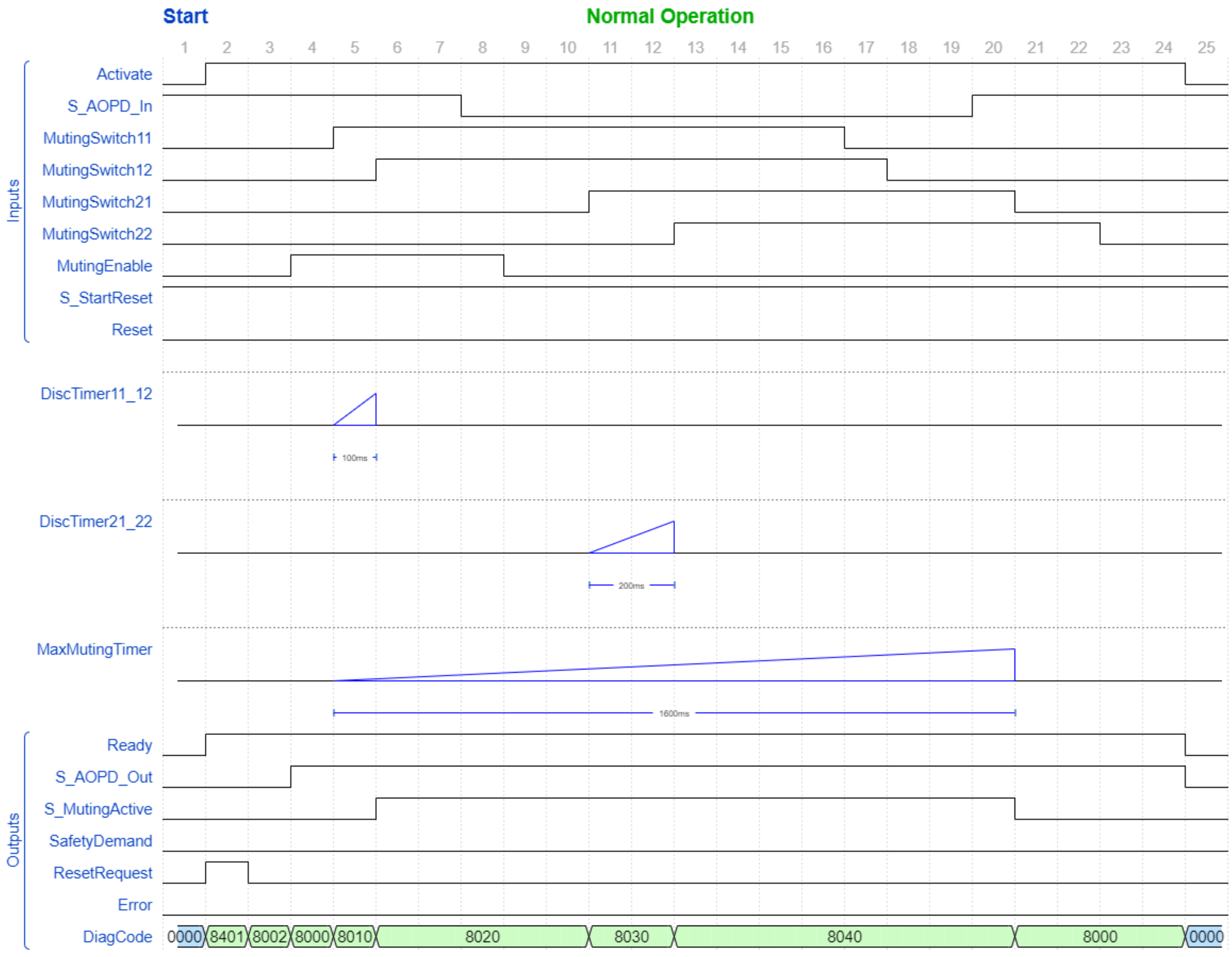


Figure 48: Timing diagram for SF_MutingPar with S_StartReset = TRUE

6.12.4 Error Detection

The FB detects the following error conditions:

- DiscTime11_12 and DiscTime21_22 have been set to values less than T#0s or greater than T#4s.
- MaxMutingTime has been set to a value less than T#0s or greater than T#120min.
- The discrepancy time for the MutingSwitch11/MutingSwitch12 or MutingSwitch21/MutingSwitch22 sensor pairs has been exceeded.
- The muting function (S_MutingActive = TRUE) exceeds the maximum muting time MaxMutingTime.
- Muting sensors MutingSwitch11, MutingSwitch12, MutingSwitch21, and MutingSwitch22 are activated in the wrong order.
- Muting sequence starts without being enabled by MutingEnable.
- A static Reset condition is detected in state 8401 and 8402.

6.12.5 Error Behavior

In the event of an error, the S_AOPD_Out and S_MutingActive outputs are set to FALSE. The DiagCode output indicates the relevant error code and the Error output is set to TRUE.
A restart is inhibited until the error conditions are cleared and the Safe state is acknowledged with Reset by the operator.

6.12.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C001	Reset Error 1	Static Reset condition detected after FB activation in state 8401.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C011	Reset Error 2	Static Reset condition detected in state 8402.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

CYx4	Error Muting sequence	Error detected in muting sequence state 8000, 8010, 8310, 8020, 8040, 8030, 8330, 8110, 8410, 8120, 8140, 8130 or 8430. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE Y = Status in the sequence (6 states for forward and 6 states for backward direction). C0x4 = Error occurred in state 8000 C1x4 = Error occurred in state Forward 8010 C2x4 = Error occurred in state Forward 8310 C3x4 = Error occurred in state Forward 8020 C4x4 = Error occurred in state Forward 8030 C5x4 = Error occurred in state Forward 8330 C6x4 = Error occurred in state Forward 8040 C7x4 = Error occurred in state Backward 8110 C8x4 = Error occurred in state Backward 8410 C9x4 = Error occurred in state Backward 8120 CAx4 = Error occurred in state Backward 8130 CBx4 = Error occurred in state Backward 8430 CCx4 = Error occurred in state Backward 8140 CFx4 = Muting Enable missing x = Status of the sensors when error occurred (4 bits: LSB = MS_11; MS_12; MS_21; MSB = MS_22).

C010	Parameter Error	DiscTime11_12, DiscTime21_22 or MaxMutingTime value out of range.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C020	Error Timer MaxMuting	Timing error: Active muting time (when S_MutingActive = TRUE) exceeds MaxMutingTime.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

C030	Error Timer MS11_12	Timing error: Discrepancy time for switching MutingSwitch11 and MutingSwitch12 > DiscTime11_12.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C040	Error Timer MS21_22	Timing error: Discrepancy time for switching MutingSwitch21 and MutingSwitch22 > DiscTime21_22.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

FB-specific status codes (no error):

Diag-Code	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8000	AOPD Free	Muting not active and no safety demand from AOPD. If timers from subsequent muting are still running, they are stopped.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8401	Init	Function block has been activated.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

8802	Safety Demand AOPD	Safety demand detected by AOPD, muting not active.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8402	Wait for Reset	Safety demand or errors have been detected and are now cleared. Operator acknowledgment by Reset required.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

Diag-Code	State Name	State Description and Output Setting

8002	Safe	Safety function activated.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8010	Muting Forward Start 1	Muting forward sequence is in starting phase after rising trigger of MutingSwitch 11. Monitoring of DiscTime11_12 is activated. Monitoring of MaxMutingTime is activated.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8310	Muting Forward Start 2	Muting forward sequence is in starting phase after rising trigger of MutingSwitch 12. Monitoring of DiscTime11_12 is activated. Monitoring of MaxMutingTime is activated.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8020	Muting Forward Active 1	Muting forward sequence is active either:
		- After rising trigger of the second entry MutingSwitch 12 or 11 has been detected.
		- When both MutingSwitch 11 and 12 have been actuated in the same cycle.
		Monitoring of DiscTime11_12 is stopped. Monitoring of MaxMutingTime is activated, when transition came directly from state 8000.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8030	Muting Forward Step 1	Muting forward sequence is active. MutingSwitch21 is the first exit switch actuated. Monitoring of DiscTime21_22 is started.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8330	Muting Forward Step 2	Muting forward sequence is active. MutingSwitch22 is the first exit switch actuated. Monitoring of DiscTime21_22 is started.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

Diag-Code	State Name	State Description and Output Setting

8040	Muting Forward Active 2	Muting forward sequence is still active. Both MutingSwitch21 and 22 are actuated, the monitoring of DiscTime21_22 is stopped.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8110	Muting Backward Start 1	Muting backward sequence is in starting phase after rising trigger of MutingSwitch21. Monitoring of DiscTime21_22 is activated. Monitoring of MaxMutingTime is activated.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8410	Muting Backward Start 2	Muting backward sequence is in starting phase after rising trigger of MutingSwitch22. Monitoring of DiscTime21_22 is activated. Monitoring of MaxMutingTime is activated.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8120	Muting Backward Active 1	Muting backward sequence is active either:
		- After rising trigger of the second MutingSwitch 21 or 22 has been detected.
		- When both MutingSwitch 21 and 22 have been actuated in the same cycle.
		Monitoring of DiscTime21_22 is stopped. Monitoring of MaxMutingTime is activated when transition came directly from state 8000.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8130	Muting Backward Step 1	Muting backward sequence is active. MutingSwitch11 is the first exit switch actuated. Monitoring of DiscTime11_12 is started.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8430	Muting Backward Step 2	Muting backward sequence is active. MutingSwitch12 is the first exit switch actuated. Monitoring of DiscTime11_12 is started.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

Diag-Code	**State Name**	**State Description and Output Setting**

8140	Muting Backward Active 2	Muting backward sequence is still active. Both exit switches MutingSwitch11 and 12 are actuated, the monitoring of DiscTime11_12 is stopped.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

6.13 Parallel Muting with 2 Sensors

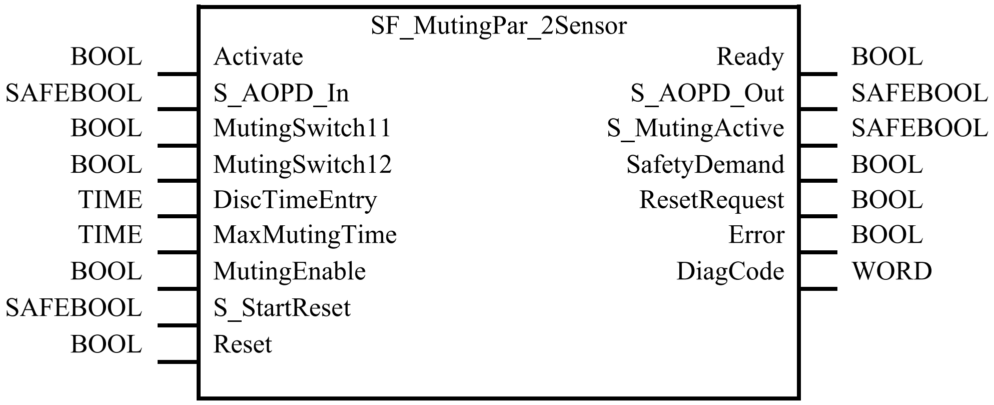
6.13.1 Applicable Safety Standards

Standards	Requirements
IEC 61496-1:2012	A.7 Muting, A.7.1.2 There shall be at least two independent hard-wired muting signal sources to initiate the function. It shall not be possible to initiate muting when the OSSDs are already in the OFF state. A.7.1.3 The mute function shall only be initiated by the correct sequence and/or timing of the mute signals. Should conflicting muting signals occur, the ESPE shall not allow a muted condition to occur. A.7.1.4 There shall be at least two independent hard-wired muting signal sources to stop the function. The muting function shall stop when the first of these muting signals changes state. The deactivation of the muting function shall not rely only on the clearance of the ESPE. A.7.1.5 The muting signals should be continuously present during muting. When the signals are not continuously present, an incorrect sequence and/or the expiration of a pre-set time limit shall cause either a lock-out condition or a restart interlock. A.7.4 Indication: A mute status signal or indicator shall be provided (in some applications, an indication signal of muting is necessary
IEC / TS 62046/Ed. 2: 2008	5.5. General Application Requirements for Muting
EN ISO 13849-1:2015	5.2.2 Manual reset function
ISO 12100: 2010	6.2.11.4: Restart following power failure/spontaneous restart

6.13.2 Interface Description

FB Name	SF_MutingPar_2Sensor		
Muting is the intended suppression of the safety function. In this FB, parallel muting with two muting sensors is specified.			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
S_AOPD_In	SAFEBOOL	FALSE	Variable. OSSD signal from AOPD. FALSE: Protection field interrupted. TRUE: Protection field not interrupted.
MutingSwitch11	BOOL	FALSE	Variable. Status of Muting sensor 11. FALSE: Muting sensor 11 not actuated. TRUE: Workpiece actuates muting sensor 11. Depending on the safety requirements it can be necessary to connect a SAFEBOOL.
MutingSwitch12	BOOL	FALSE	Variable. Status of Muting sensor 12. FALSE: Muting sensor 12 not actuated. TRUE: Workpiece actuates muting sensor 12. Depending on the safety requirements it can be necessary to connect a SAFEBOOL.
DiscTimeEntry	TIME	T#0s	Constant 0..4 s; Max. discrepancy time for S_MutingSwitch11 and S_MutingSwitch12 entering muting gate
MaxMutingTime	TIME	T#0s	Constant 0..120 min; Maximum time for complete muting sequence, timer starts with the first actuated muting sensor.

MutingEnable	BOOL	FALSE	Variable or constant. Command by the control system that enables the start of the muting function when needed by the machine cycle. After the start of the muting function, this signal can be switched off. FALSE: Muting not enabled TRUE: Start of Muting function enabled
S_StartReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
Reset	BOOL	FALSE	See Section 5.1.1 General Input Parameters
VAR_OUTPUT			
Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
S_AOPD_Out	SAFEBOOL	FALSE	Safety related output indicates status of the muted guard. FALSE: AOPD protection field interrupted and muting not active. TRUE: AOPD protection field not interrupted or muting active.
S_MutingActive	SAFEBOOL	FALSE	Indicates status of Muting process. FALSE: Muting not active. TRUE: Muting active.
SafetyDemand	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
ResetRequest	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters
Notes: Line control of muting sensor signals must be active in the safety loop			



6.13.3 Functional Description

Muting is the intended suppression of the safety function. This is required, e.g., when transporting the material into the danger zone without causing the machine to stop. Muting is triggered by muting sensors. The use of two muting sensors and correct integration into the production sequence must ensure that no persons enter the danger zone while the light curtain is muted. Muting sensors can be pushbuttons, proximity switches, photoelectric barriers, limit switches, etc. which do not have to be failsafe. Active muting mode must be indicated by indicator lights.

There are sequential and parallel muting procedures. In this FB, parallel muting with two muting sensors was used; an explanation is provided below. The positioning of the sensors should be as described in Annex F.7 of IEC 62046, CD 2005, as shown in Figure 49. The FB can be used in both directions, forward and backward. However, the actual direction cannot be identified. The muting should be enabled with the MutingEnable signal by the process control to avoid manipulation.

The FB input parameters include the signals of the two muting sensors (MutingSwitch11 and MutingSwitch12), the OSSD signal from the "active opto-electronic protective device", S_AOPD_In, as well as two parameterizable times (DiscTimeEntry and MaxMutingTime).

The S_StartReset input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

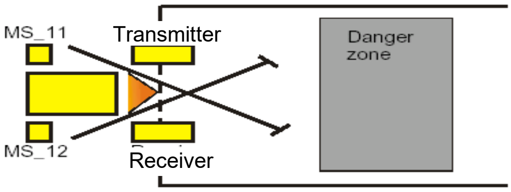
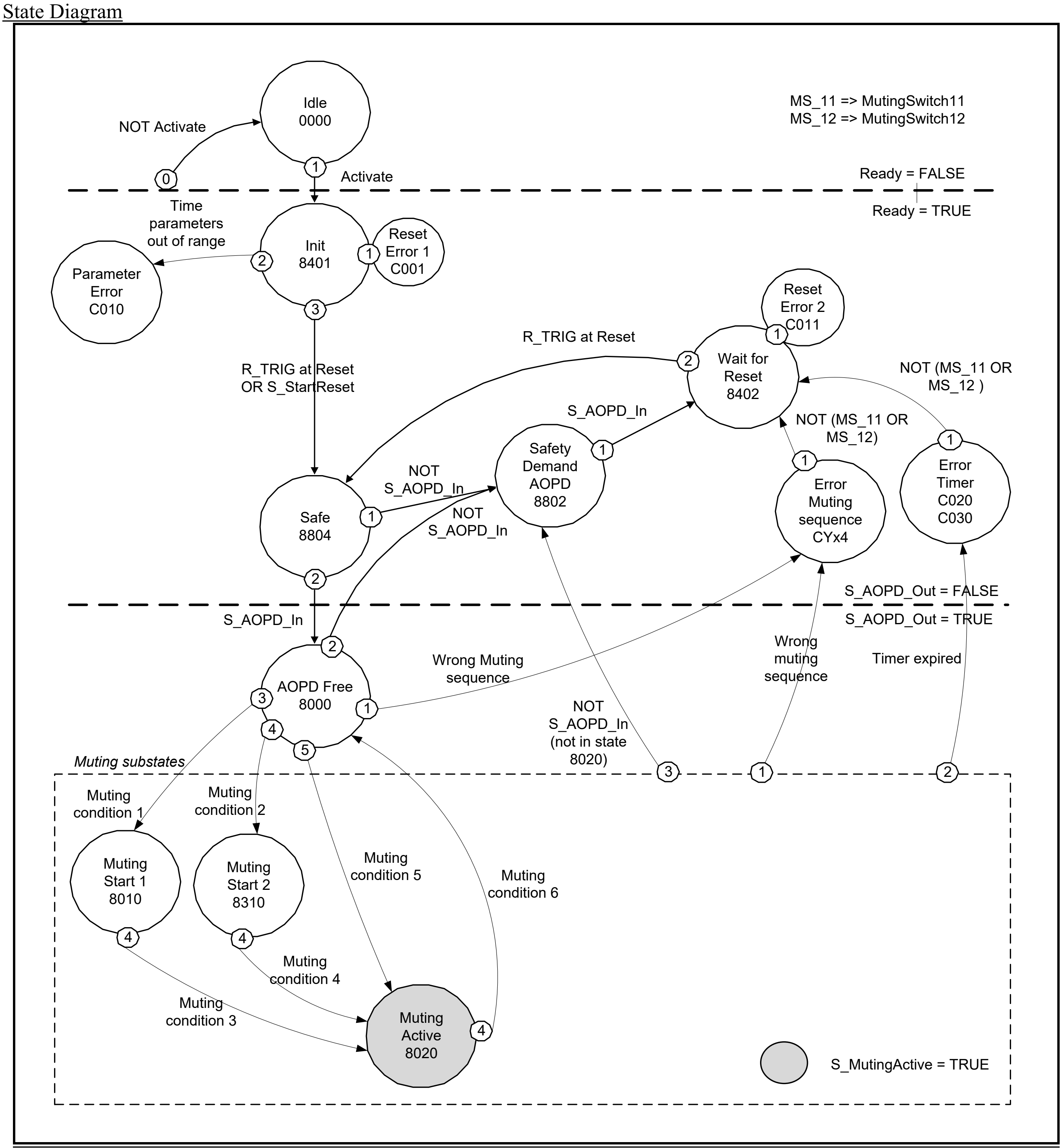
No.	Figure	Explanation
1		If reflection light barriers are used as muting sensors, they are generally arranged diagonally. In general, this arrangement of reflection light barriers as muting sensors requires only two light barriers, and only MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are allocated.

Figure 49: Example for SF_MutingPar_2Sensor with two reflecting light barriers

State Diagram



Note1: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Note 2: Within muting substates, transitions due to Error Muting sequence (priority 1), Error Timer (priority 2), Safety demand AOPD (priority 3) have higher priority than transitions to Muting substates (priority 4).

Note 3: Muting conditions are defined below.

Figure 50: State diagram for SF_MutingPar_2Sensor

Muting conditions:

Muting condition 1 (to 8010) (MS_11 is the first entry switch actuated). Start timer DiscTimeEntry and MaxMutingTime**:** MutingEnable **AND R_TRIG at** MS_11 **AND NOT** MS_12
Muting condition 2 (to 8310) (MS_12 is the first entry switch actuated). Start timer DiscTimeEntry and MaxMutingTime**:** MutingEnable **AND NOT** MS_11 **AND R_TRIG at** MS_12
Muting condition 3 (from 8010 to 8020) (MS_12 is the second entry switch actuated)**:** Stop timer DiscTimeEntry: MutingEnable **AND** MS_11 **AND R_TRIG at** MS_12
Muting condition 4 (from 8310 to 8020) (MS_11 is the second entry switch actuated)**:** Stop timer DiscTimeEntry: MutingEnable **AND R_TRIG at** MS_11 **AND** MS_12
Muting condition 5 (from 8000 to 8020) (both switches actuated in same cycle)**:** Start Timer MaxMutingTime: MutingEnable **AND R_TRIG at** MS_11 **AND R_TRIG at** MS_12
Muting condition 6 (from 8020 to 8000) (both switches released in same cycle or MS_11 and MS_12 released consecutively). Stop timer MaxMutingTime**: NOT** MS_11 **OR NOT** MS_12

Wrong 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 8010: **NOT** MutingEnable **OR NOT** MS_11
State 8310: **NOT** MutingEnable **OR NOT** MS_12
State 8020: No case of wrong muting sequences in this state (prio 1).

Typical Timing Diagram

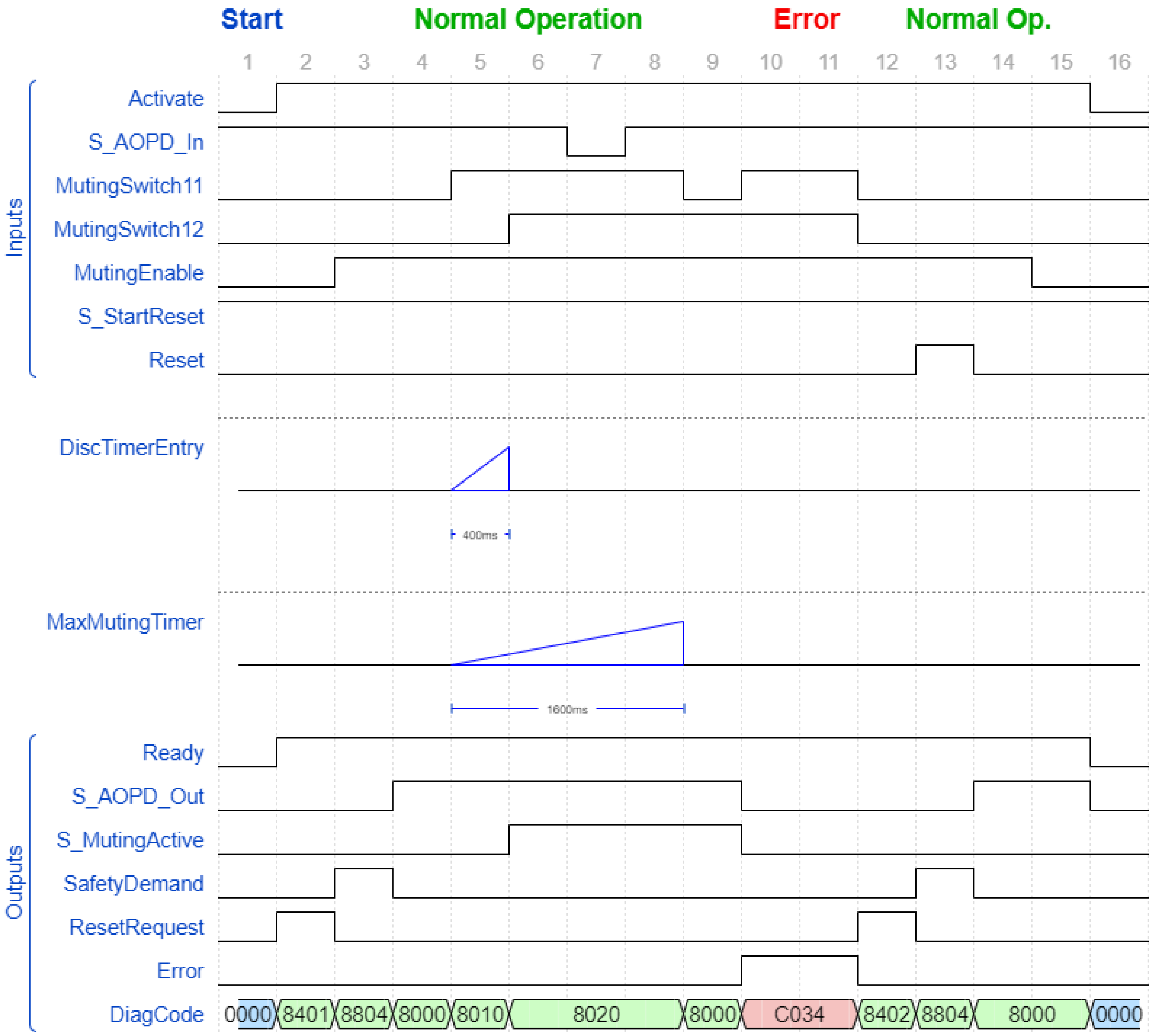


Figure 51: Timing diagram for SF_MutingPar_2Sensor (S_StartReset = TRUE, Reset = FALSE,)

6.13.4 Error Detection

The FB detects the following error conditions:

- DiscTimeEntry has been set to value less than T#0s or greater than T#4s.
- MaxMutingTime has been set to a value less than T#0s or greater than T#120min.
- The discrepancy time for the MutingSwitch11/ MutingSwitch12 sensor pair has been exceeded.
- The muting function (MutingActive = TRUE) exceeds the maximum muting time MaxMutingTime.
- Muting sensors MutingSwitch11, MutingSwitch12 are activated in the wrong order.
- Muting sequence starts without being enabled by MutingEnable.
- Static muting sensor signals.
- A static Reset condition is detected in state 8401 and 8402.

6.13.5 Error Behavior

In the event of an error, the S_AOPD_Out and S_MutingActive outputs are set to FALSE. The Error output is set to TRUE and the DiagCode output indicates the relevant error code.
A restart is inhibited until the error conditions are cleared and the Safe state is acknowledged with Reset by the operator.

6.13.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C001	Reset Error 1	Static Reset condition detected after FB activation in state 8401.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C011	Reset Error 2	Static Reset condition detected in state 8402.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

CYx4	Error Muting sequence	Error detected in muting sequence state 8000, 8010, 8310.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE
		Y = Status in the sequence
		C0x4 = Error occurred in state 8000
		C1x4 = Error occurred in state 8010
		C2x4 = Error occurred in state 8310
		
		CFx4 = Muting Enable missing
		x = Status of the sensors when error occurred (4 bits: LSB = MS_11; next to LSB = MS_12).

C010	Parameter Error	DiscTimeEntry or MaxMutingTime value out of range.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

C020	Error timer MaxMuting	Timing error: Active muting time (when S_MutingActive = TRUE) exceeds MaxMutingTime.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C030	Error timer Entry	Timing error: Discrepancy time for switching MutingSwitch11 and MutingSwitch12 from FALSE to TRUE > DiscTimeEntry.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8000	AOPD Free	Muting not active and no safety demand from AOPD. If timers from subsequent muting are still running, they are stopped.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8401	Init	Function block was activated.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

8802	Safety Demand AOPD	Safety demand detected by AOPD, muting not active.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8402	Wait for Reset	Safety demand or errors have been detected and are now cleared. Operator acknowledgment by Reset required.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8804	Safe	Safety function activated.
		Ready = TRUE
		S_AOPD_Out = FALSE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8010	Muting Start 1	Muting sequence is in starting phase after rising trigger of Muting-Switch11. Monitoring of DiscTimeEntry is activated.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8310	Muting Start 2	Muting sequence is in starting phase after rising trigger of Muting-Switch12. Monitoring of DiscTimeEntry is activated.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8020	Muting Active	Muting sequence is active either:
		- After rising trigger of the second MutingSwitch 12 or 11 has been detected.
		- When both MutingSwitch 11 and 12 have been actuated in the same cycle.
		Monitoring of DiscTimeEntry is stopped. Monitoring of MaxMutingTime is activated.
		Ready = TRUE
		S_AOPD_Out = TRUE
		S_MutingActive = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

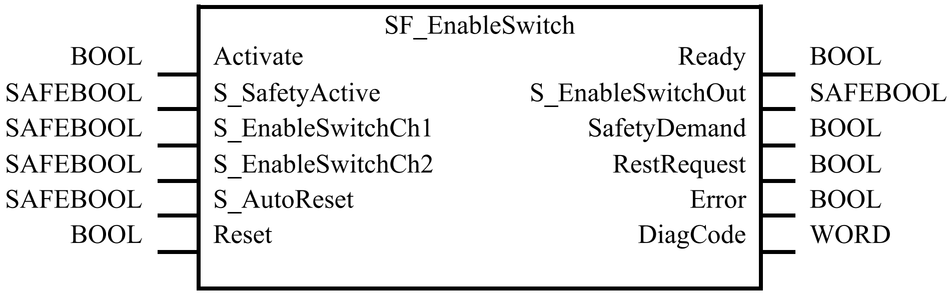
6.14 Enable Switch

6.14.1 Applicable Safety Standards

Standards	Requirements
IEC 60204-1:2016	9.2.6.3: Enabling control (see also 10.9) is a manually activated control function interlock that: a) when activated allows a machine operation to be initiated by a separate start control, and b) when de-activated – initiates a stop function, and – prevents initiation of machine operation. Enabling control shall be so arranged as to minimize the possibility of defeating, for example by requiring the de-activation of the enabling control device before machine operation may be reinitiated. It should not be possible to defeat the enabling function by simple means. 10.9: When an enabling control device is provided as a part of a system, it shall signal the enabling control to allow operation when actuated in one position only. In any other position, operation shall be stopped or prevented. Enabling control devices shall be selected that have the following features: … – for a three-position type: - position 1: off-function of the switch (actuator is not operated). - position 2: enabling function (actuator is operated in its mid position). - position 3: off-function (actuator is operated past its mid position). - when returning from position 3 to position 2, the enabling function is not activated.
EN ISO 13849-1:2015	5.2.2 Manual reset function
ISO 12100-2: 2010	4.11.4: Restart following power failure/spontaneous restart

6.14.2 Interface Description

FB Name	**SF_EnableSwitch**			
The SF_EnableSwitch FB evaluates the signals of an enable switch with three positions.				
VAR_INPUT				
	Name	*Data Type*	*Initial Value*	*Description, parameter values*
	Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
	S_SafetyActive	SAFEBOOL	FALSE	Variable or constant. Confirmation of the safe mode (limitation of the speed or the power of motion, limitation of the range of motion). FALSE: Safe mode is not active. TRUE: Safe mode is active.
	S_EnableSwitchCh1	SAFEBOOL	FALSE	Variable. Signal of contacts E1 and E2 of the connected enable switch. FALSE: Connected switches are open. TRUE: Connected switches are closed.
	S_EnableSwitchCh2	SAFEBOOL	FALSE	Variable. Signal of contacts E3 and E4 of the connected enable switch. FALSE: Connected switches are open. TRUE: Connected switches are closed.
	S_AutoReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
	Reset	BOOL	FALSE	See Section 5.1.1 General Input Parameters
VAR_OUTPUT				
	Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
	S_EnableSwitchOut	SAFEBOOL	FALSE	Safety related output: Indicates suspension of guard. FALSE: Disable suspension of safeguarding. TRUE: Enable suspension of safeguarding.
	SafetyDemand	BOOL	FALSE	See 5.1.2 General Output Parameters
	ResetRequest	BOOL	FALSE	See 5.1.2 General Output Parameters
	Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
	DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters
Notes: -				



6.14.3 Functional Description

The SF_EnableSwitch FB supports the suspension of safeguarding (DIN EN 60204 Section 9.2.4) using enable switches (DIN EN 60204 Section 9.2.5.8), if the relevant operating mode is selected and active. The relevant operating mode (limitation of the speed or the power of motion, limitation of the range of motion) must be selected outside the SF_EnableSwitch FB.
The SF_EnableSwitch FB evaluates the signals of an enable switch with three positions (DIN EN 60204 Section 9.2.5.8).
The S_EnableSwitchCh1 and S_EnableSwitchCh2 input parameters process the following signal levels of contacts E1 to E4:

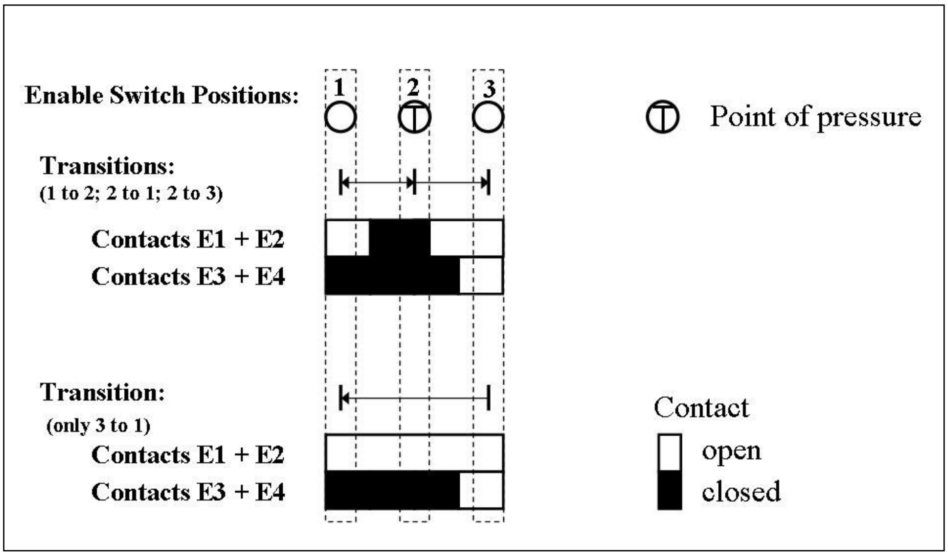


Figure 52: Switch positions

The signal from E1+E2 must be connected to the S_EnableSwitchCh1 parameter. The signal from E3+E4 must be connected to the S_EnableSwitchCh2 parameter. The position of the enable switch is detected in the FB using this signal sequence.
The transition from position 2 to 3 can be different from shown here.

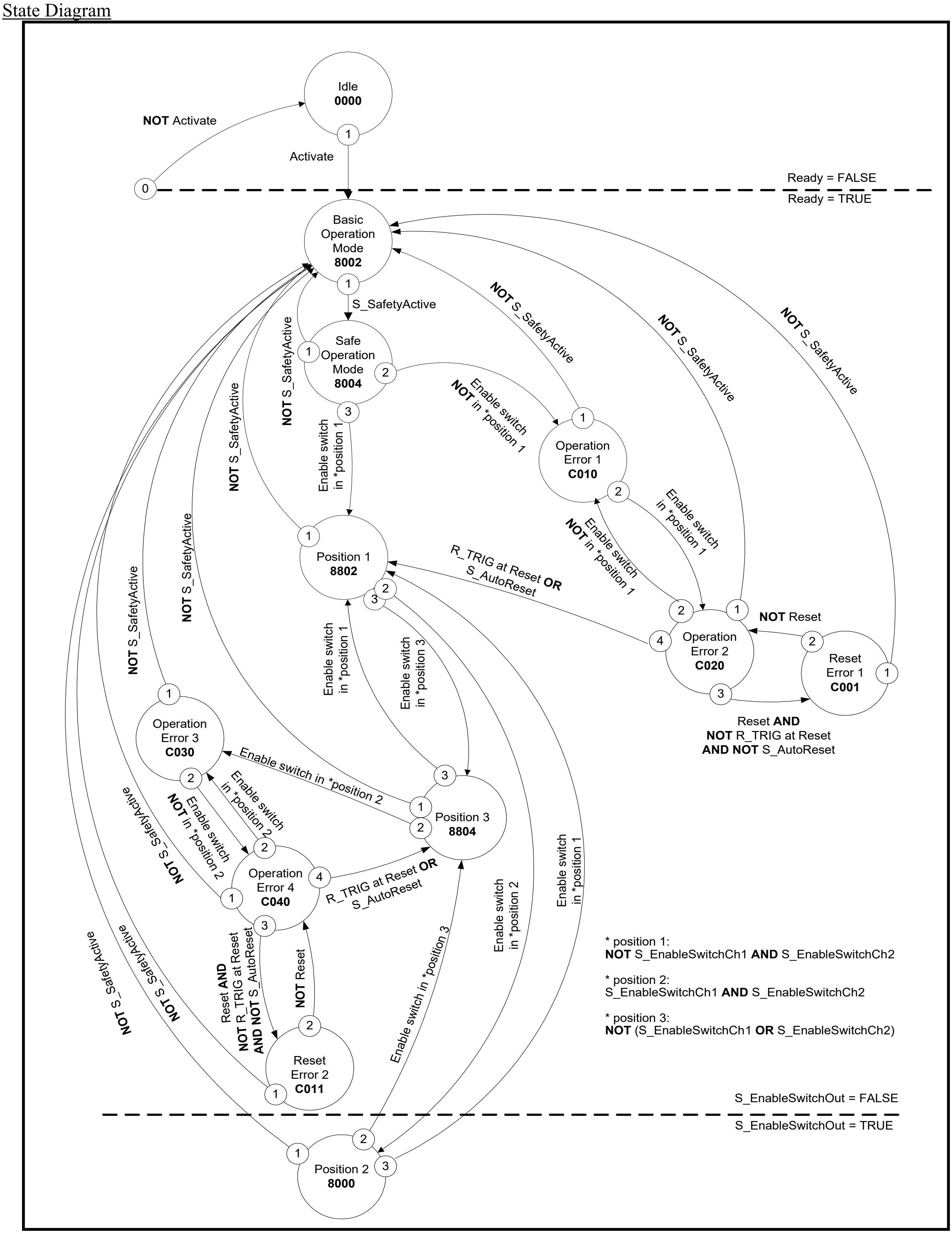
The switching direction (position 1 => position 2/position 3 => position 2) can be detected in the FB using the defined signal sequence of the enable switch contacts. The suspension of safeguarding can only be enabled by the FB after a move from position 1 to position 2. Other switching directions or positions may not be used to enable the suspension of safeguarding. This measure meets the requirements of EN 60204 Section 9.2.5.8.

In order to meet the requirements of DIN EN 60204 Section 9.2.4, the user shall use a suitable switching device. In addition, the user must ensure that the relevant operating mode (DIN EN 60204 Section 9.2.3) is selected in the application (automatic operation must be disabled in this operating mode using appropriate measures).

The operating mode is usually specified using an operating mode selection switch in conjunction with the SF_ModeSelector FB and the SF_SafeRequest or SF_SafelyLimitedSpeed FB.

The SF_EnableSwitch FB processes the confirmation of the "safe mode" state via the "S_SafetyActive" parameter. On implementation in an application of the safe mode without confirmation, a static TRUE signal is connected to the "S_SafetyActive" parameter.
The S_AutoReset input shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

State Diagram

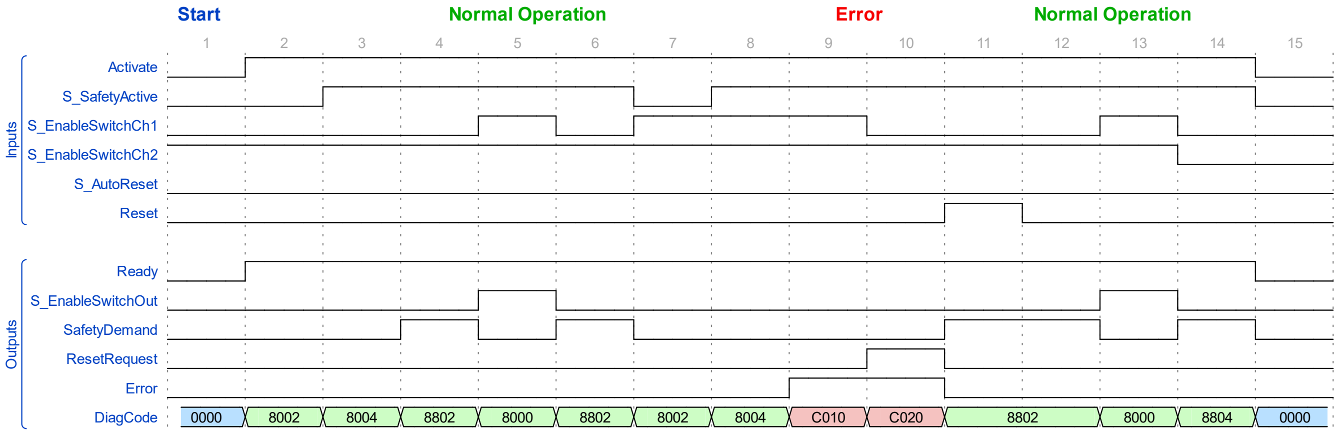

Idle
0000
1
NOT Activate
Activate
0
Ready = FALSE
Ready = TRUE
Basic Operation Mode
8002
1
S_SafetyActive
Safe Operation Mode
8004
1
2
3
NOT S_SafetyActive
NOT S_SafetyActive
NOT S_SafetyActive
NOT S_SafetyActive
Enable switch NOT in *position 1
Operation Error 1
C010
1
2
Enable switch in *position 1
Enable switch NOT in *position 1
Enable switch in *position 1
NOT S_SafetyActive
NOT S_SafetyActive
NOT S_SafetyActive
Position 1
8802
1
2
3
R_TRIG at Reset OR S_AutoReset
Operation Error 2
C020
1
2
3
4
NOT Reset
Reset Error 1
C001
1
2
Reset AND NOT R_TRIG at Reset AND NOT S_AutoReset
Enable switch in *position 1
Enable switch in *position 3
Operation Error 3
C030
1
2
Enable switch in *position 2
Enable switch in *position 2
NOT S_SafetyActive
Enable switch NOT in *position 2
Position 3
8804
1
2
3
Operation Error 4
C040
1
2
3
4
R_TRIG at Reset OR S_AutoReset
Reset AND NOT R_TRIG at Reset AND NOT S_AutoReset
NOT Reset
Reset Error 2
C011
1
2
Enable switch in *position 3
Enable switch in *position 2
Enable switch in *position 1
NOT S_SafetyActive
NOT S_SafetyActive
S_EnableSwitchOut = FALSE
S_EnableSwitchOut = TRUE
Position 2
8000
1
2
3
* position 1:
NOT S_EnableSwitchCh1 AND S_EnableSwitchCh2
* position 2:
S_EnableSwitchCh1 AND S_EnableSwitchCh2
* position 3:
NOT (S_EnableSwitchCh1 OR S_EnableSwitchCh2)

Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Figure 53: State diagram for SF_EnableSwitch

Typical Timing Diagrams

S_AutoReset = FALSE



S_AutoReset = TRUE

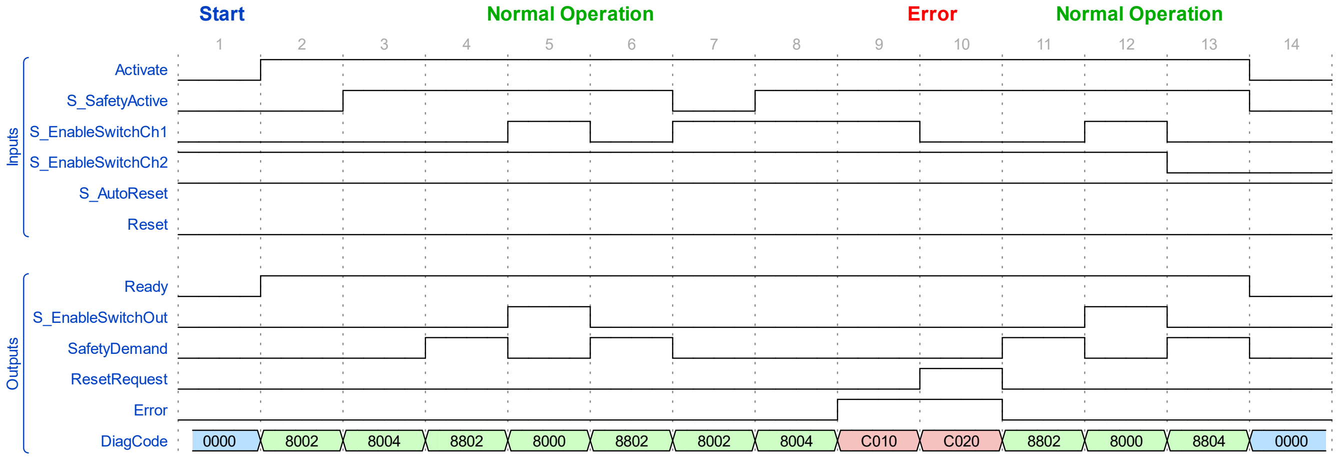


Figure 54: Timing diagram for SF_EnableSwitch.

6.14.4 Error Detection

The following conditions force a transition to the Error state:

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

6.14.5 Error Behavior

In the event of an error, the S_EnableSwitchOut safe output is set to FALSE and remains in this Safe state.
Different from other FBs, a Reset Error state can be left by the condition Reset = FALSE or, additionally, when the signal S_SafetyActive is FALSE.
Once the error has been removed, the enable switch must be in the initial position specified in the process before the S_EnableSwitchOut output can be set to TRUE using the enable switch. If S_AutoReset = FALSE, a rising trigger is required at Reset.

6.14.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C001	Reset Error 1	Static Reset signal detected in state C020.
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

C011	Reset Error 2	Static Reset signal detected in state C040.
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C010	Operation Error 1	Enable switch not in position 1 during activation of S_SafetyActive.
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C020	Operation Error 2	Enable switch in position 1 after C010.
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE

C030	Operation Error 3	Enable switch in position 2 after position 3.
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C040	Operation Error 4	Enable switch not in position 2 after C030.
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_EnableSwitchOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8002	Basic Operation Mode	Safe operation mode is not active.
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8004	Safe Operation Mode	Safe operation mode is active.
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8802	Position 1	Safe operation mode is active and the enable switch is in position 1.
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8804	Position 3	Safe operation mode is active and the enable switch is in position 3.
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8000	Position 2	Safe operation mode is active and the enable switch is in position 2.
		Ready = TRUE
		S_EnableSwitchOut = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

6.15 EnableSwitch 2 (without detection of panic position)

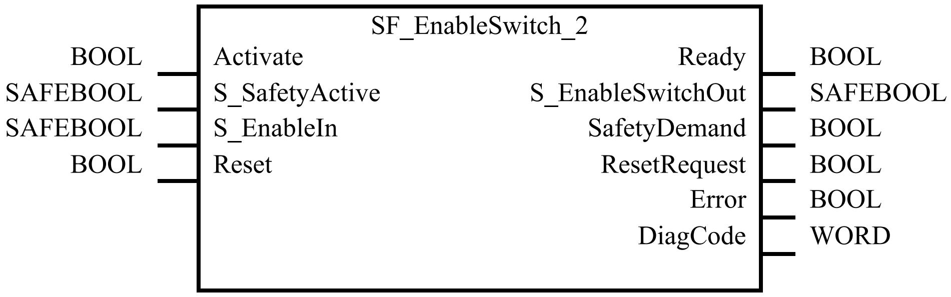
6.15.1 Applicable Safety Standards

Standards	Requirements
IEC 60204-1,: 2016	9.2.6.3: Enabling control (see also 10.9) is a manually activated control function interlock that: a) when activated allows a machine operation to be initiated by a separate start control, and b) when de-activated – initiates a stop function, and – prevents initiation of machine operation. Enabling control shall be so arranged as to minimize the possibility of defeating, for example by requiring the de-activation of the enabling control device before machine operation may be reinitiated. It should not be possible to defeat the enabling function by simple means. 10.9: When an enabling control device is provided as a part of a system, it shall signal the enabling control to allow operation when actuated in one position only. In any other position, operation shall be stopped or prevented. Enabling control devices shall be selected that have the following features: … – for a two-position type: - position 1: off-function of the switch (actuator is not operated). - position 2: enabling function (actuator is operated). – for a three-position type: - position 1: off-function of the switch (actuator is not operated). - position 2: enabling function (actuator is operated in its mid position). - position 3: off-function (actuator is operated past its mid position). - when returning from position 3 to position 2, the enabling function is not activated.
EN ISO 13849-1:2015	5.2.2 Manual reset function. <Note: a positive edge evaluation has the same quality as a negative edge evaluation>
ISO 12100: 2010	6.2.11.4 Restart after power interruption If a hazard could be generated, the spontaneous restart of a machine when it is re-energized after power interruption shall be prevented (for example, by use of a self-maintained relay, contactor or valve).

Note: Many three position switches are wired internally and do not provide an external contact for evaluating the panic position (position 3). If a position switch is used that offers an external contact to evaluate externally the position 3, the SF_EnableSwitch shall be used.

6.15.2 Interface Description

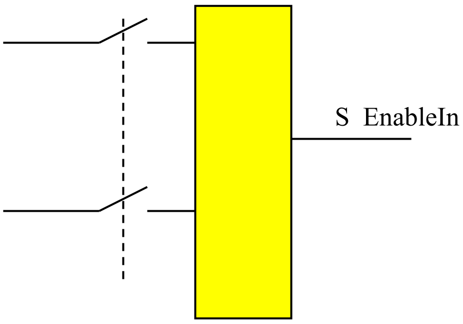
FB Name	**SF_EnableSwitch_2**		
The SF_EnableSwitch FB_2 evaluates the signals of an enable switch with two or three positions.			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, parameter values*
Activate	BOOL	FALSE	See 5.1.1 General Input Parameters
S_SafetyActive	SAFEBOOL	FALSE	Variable or constant. Confirmation of the safe mode (limitation of the speed or the power of motion, limitation of the range of motion). FALSE: Safe mode is not active. TRUE: Safe mode is active.
S_EnableIn	SAFEBOOL	FALSE	Variable. Signal of connected enable switch. The evaluation of the signals (discrepancy) will be done within the IO unit or the FB_Equivalent FALSE: Not Enabled. TRUE: Enabled.
Reset	BOOL	FALSE	See 5.1.1 General Input Parameters
VAR_OUTPUT			
Ready	BOOL	FALSE	See 5.1.2 General Output Parameters
S_EnableSwitchOut	SAFEBOOL	FALSE	Safety related output: Indicates suspension of guard. FALSE: Disable suspension of safeguarding. TRUE: Enable suspension of safeguarding.
SafetyDemand	BOOL	FALSE	See 5.1.2 General Output Parameters
ResetRequest	BOOL	FALSE	See 5.1.2 General Output Parameters
Error	BOOL	FALSE	See 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See 5.1.2 General Output Parameters
Notes: -			



6.15.3 Functional Description

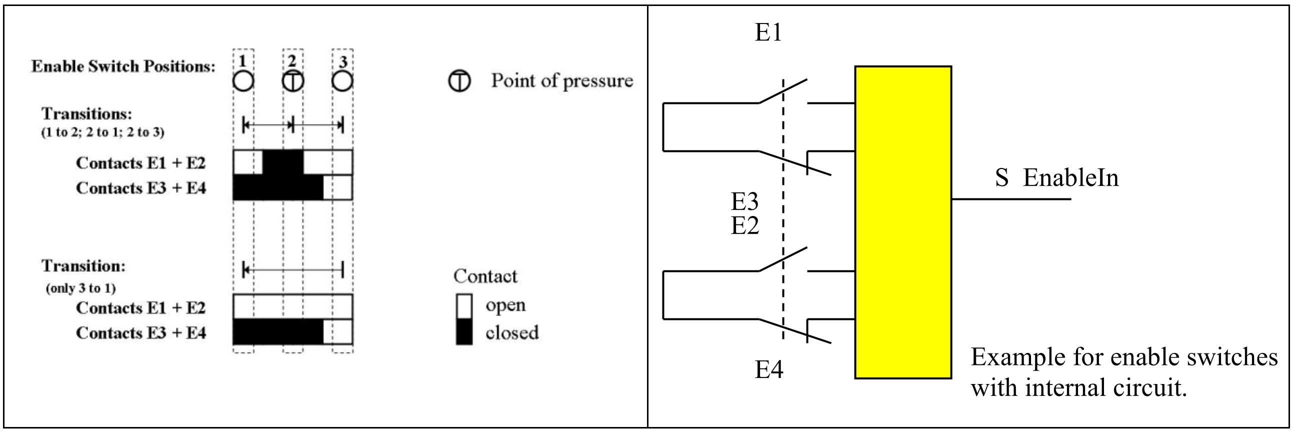
The SF_EnableSwitch_2 FB supports the suspension of safeguarding (DIN EN 60204 Section 9.2.4) using enable switches (DIN EN 60204 Section 9.2.5.8) if the relevant operating mode is selected and active. The relevant operating mode (limitation of the speed or the power of motion, limitation of the range of motion) must be selected outside the SF_EnableSwitch_2 FB.
The SF_EnableSwitch_2 FB evaluates the signals of an enable switch with two or three positions (DIN EN 60204 Section 9.2.5.8).

Two position switch



Three position switch

There is an internal circuit between the normally closed and normally open contacts as shown below. The output is either HIGH if the enable switch is in Pos 2 or LOW if either the enable switch is released (Pos1) or in the panic position (Pos3).



The suspension of safeguarding can only be enabled by the FB after a move from position 1 to position 2. Other switching directions or positions may not be used to enable the suspension of safeguarding. This measure meets the requirements of EN 60204 Section 9.2.5.8.
In order to meet the requirements of EN 60204 Section 9.2.4, the user shall use a suitable switching device. In addition, the user must ensure that the relevant operating mode (EN 60204 Section 9.2.3) is selected in the application (automatic operation must be disabled in this operating mode using appropriate measures).

The operating mode is usually specified using an operating mode selection switch in conjunction with the SF_ModeSelector FB and the SF_SafeRequest or SF_SafelyLimitedSpeed FB.

The SF_EnableSwitch FB processes the confirmation of the "safe mode" state via the "S_SafetyActive" parameter. On implementation in an application of the safe mode without confirmation, a static TRUE signal is connected to the "S_SafetyActive" parameter.

State Diagram

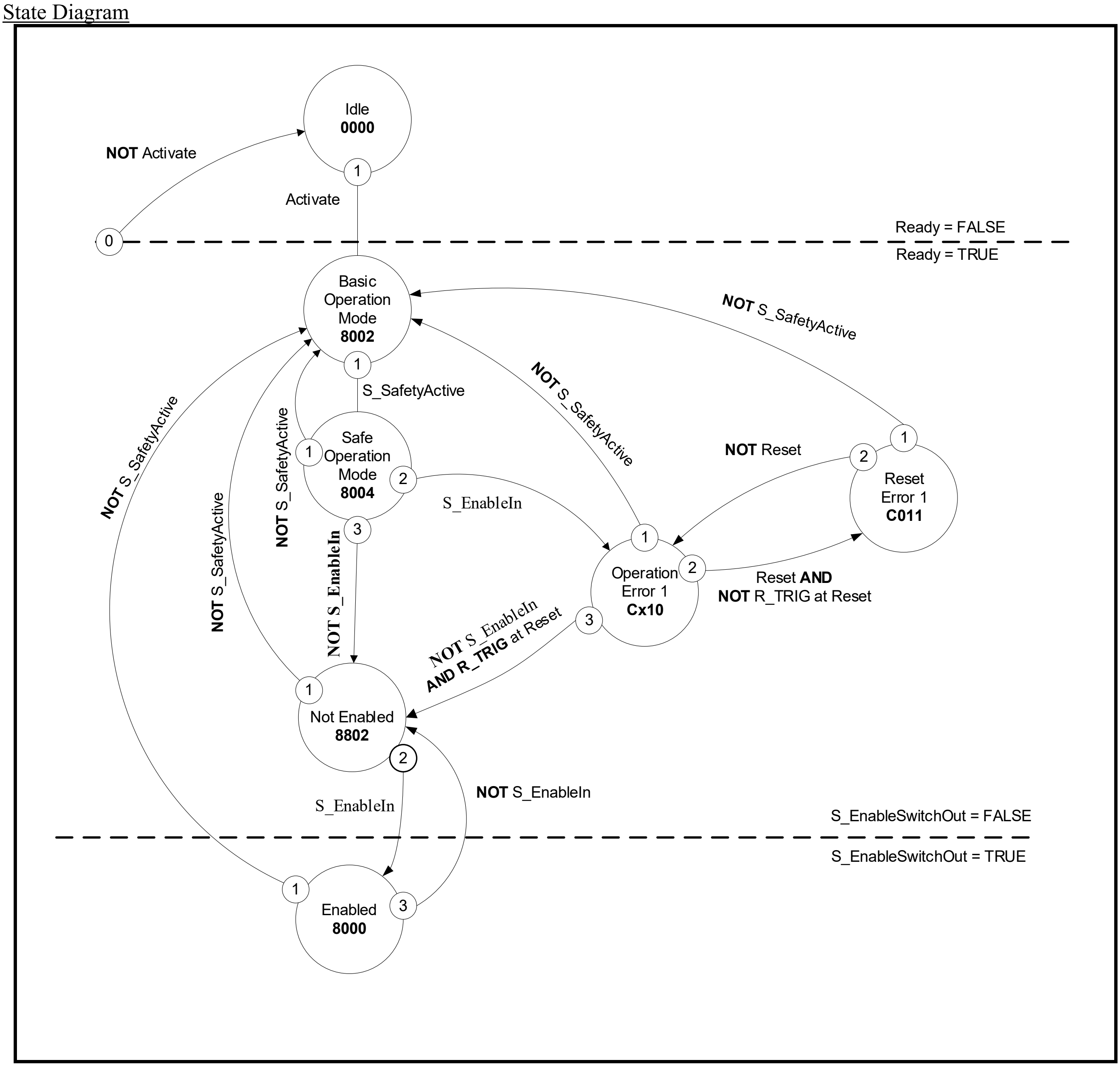


Figure 55: State diagram for SF_EnableSwitch_2

Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Typical Timing Diagrams

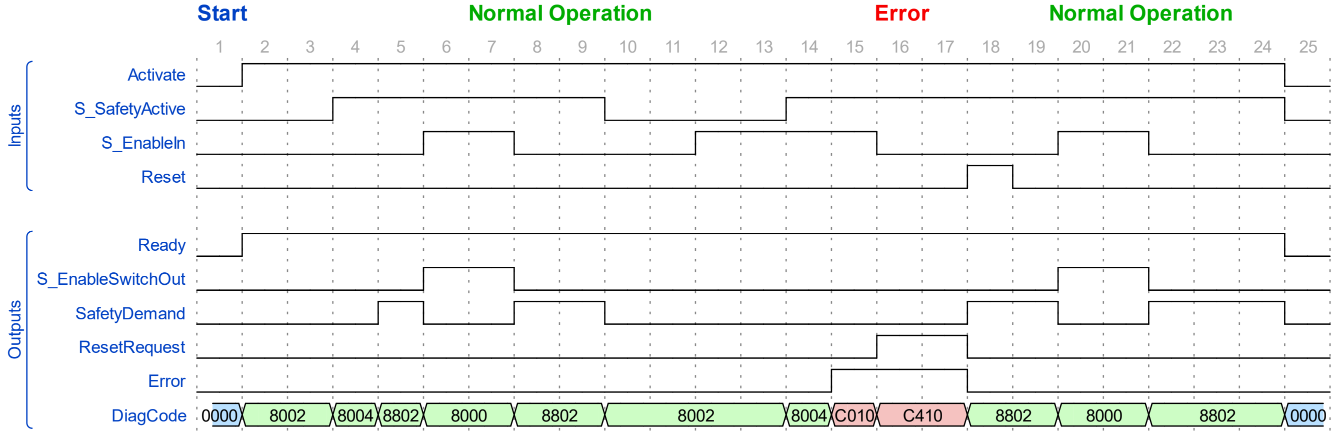


Figure 56: Timing diagram for SF_EnableSwitch_2

6.15.4 Error Detection

It will be detected if the enable Switch is already pressed when Safety becomes active. The machine must be put in a safe state first before the enable switch can be used.
In case Reset is requested, a permanent Reset signal TRUE will be detected (Reset error).

6.15.5 Error Behavior

In the event of an error, the S_EnableSwitchOut safe output is set to FALSE and remains in this Safe state. Once the S_EnableIn becomes FALSE, via releasing the enable switch by the operator, the error can be reset via the Reset input. If during the error condition TRUE, S_SafetyActive becomes FALSE, there is no need for a separate Reset. However, if the EnableSwitch is not released before S_SafetyActive becomes TRUE again, a transition to the error state C010 is made.

6.15.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C001	Reset Error 1	Static Reset signal detected in state Cx10.
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

Cx10	Operation Error 1	Enable switch not in position 1 during activation of S_SafetyActive.
		
		IF S_EnableIn = TRUE
		x = 0 ELSE x = 4
		
		Output signals for x = 0 (C010)
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE
		
		Output signals for x = 4 (C410)
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_EnableSwitchOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8002	Basic Operation Mode	Safe operation mode is not active.
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8004	Safe Operation Mode	Safe operation mode is active.
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8802	Not Enabled	Safe operation mode is active and the enable switch is in position 1 or 3.
		Ready = TRUE
		S_EnableSwitchOut = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8000	Enabled	Safe operation mode is active and the enable switch is in position 2.
		Ready = TRUE
		S_EnableSwitchOut = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

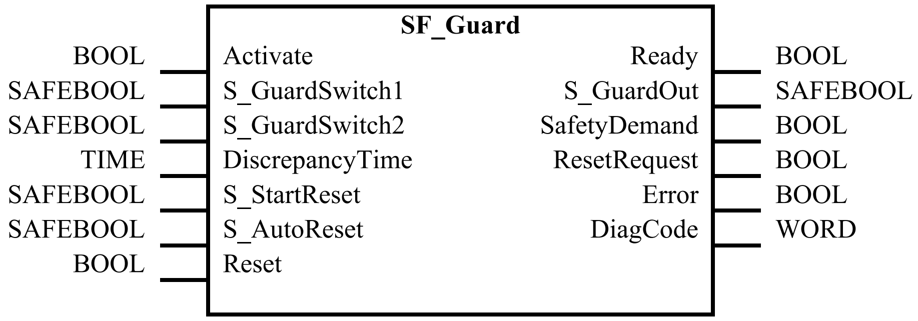
6.16 Safety Guard

6.16.1 Applicable Safety Standards

Standards	Requirements
ISO 14119: 2013	Interlocking devices associated with guards – principles for design and selection
ISO 14120: 2015	3.5 Interlocking guard - the hazardous machine functions "covered" by the guard cannot operate until the guard is closed. - if the guard is opened while hazardous machine functions are operating, a stop command is given. 3.5.1 interlocking guard with a start function control guard special form of interlocking guard which, once it has reached its closed position, gives a command to initiate the hazardous machine function(s) without the use of a separate start control
EN ISO 13849-1:2015	5.2.2 Manual reset function
ISO 12100-2: 2010	4.11.4 Restart following power failure/spontaneous restart
IEC 60947-5-3: 2013	Low voltage switchgear and control gear – Part 5.3: Control circuit devices and switching elements – Requirements for proximity devices with defined behavior under fault conditions (PDDB)

6.16.2 Interface Description

FB Name		**SF_Guard**		
This function block monitors the relevant safety guard. There are two independent input parameters for two switches at the safety guard coupled with a time difference (DiscrepancyTime) for closing the guard.				
VAR_INPUT				
	Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
	Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
	S_GuardSwitch1	SAFEBOOL	FALSE	Variable. Guard switch 1 input. FALSE: Guard is not closed. TRUE: Guard is closed.
	S_GuardSwitch2	SAFEBOOL	FALSE	Variable. Guard switch 2 input. FALSE: Guard is not closed. TRUE: Guard is closed.
	DiscrepancyTime	TIME	T#0ms	Constant. Configures the monitored synchronous time between S_GuardSwitch1 and S_GuardSwitch2.
	S_StartReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
	S_AutoReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
	Reset	BOOL	FALSE	See Section 5.1.1 General Input Parameters
VAR_OUTPUT				
	Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
	S_GuardOut	SAFEBOOL	FALSE	Output indicating that the guard is closed and the guarded area safe. FALSE: Guard is open. TRUE: both S_GuardSwitches are TRUE, no error and acknowledgment. Guard is closed.
	SafetyDemand	BOOL	FALSE	See 5.1.2 General Output Parameters
	ResetRequest	BOOL	FALSE	See 5.1.2 General Output Parameters
	Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
	DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters
Notes: • In version 1.0 the name was SF_GuardMonitoring. It was decided to change this name to SF_Guard, however the original name can be used also. • For certain (lower) levels of safety requirements it can be allowed to use BOOL as inputs and SAFEBOOL as output. However, this has to be evaluated via the FMEA of the application. In the library there should be made a distinction between the SAFEBOOL and BOOL version.				



6.16.3 Functional Description

The function block requires two inputs indicating the guard position for safety guards with two switches (according to ISO 14119), a DiscrepancyTime input and Reset input. If the safety guard only has one switch, the S_GuardSwitch1 and S_GuardSwitch2 inputs can be bridged. The discrepancy time is the maximum time required for both switches to respond when closing the safety guard. The Reset, S_StartReset, and S_AutoReset inputs determine how the function block is reset after the safety guard has been opened.

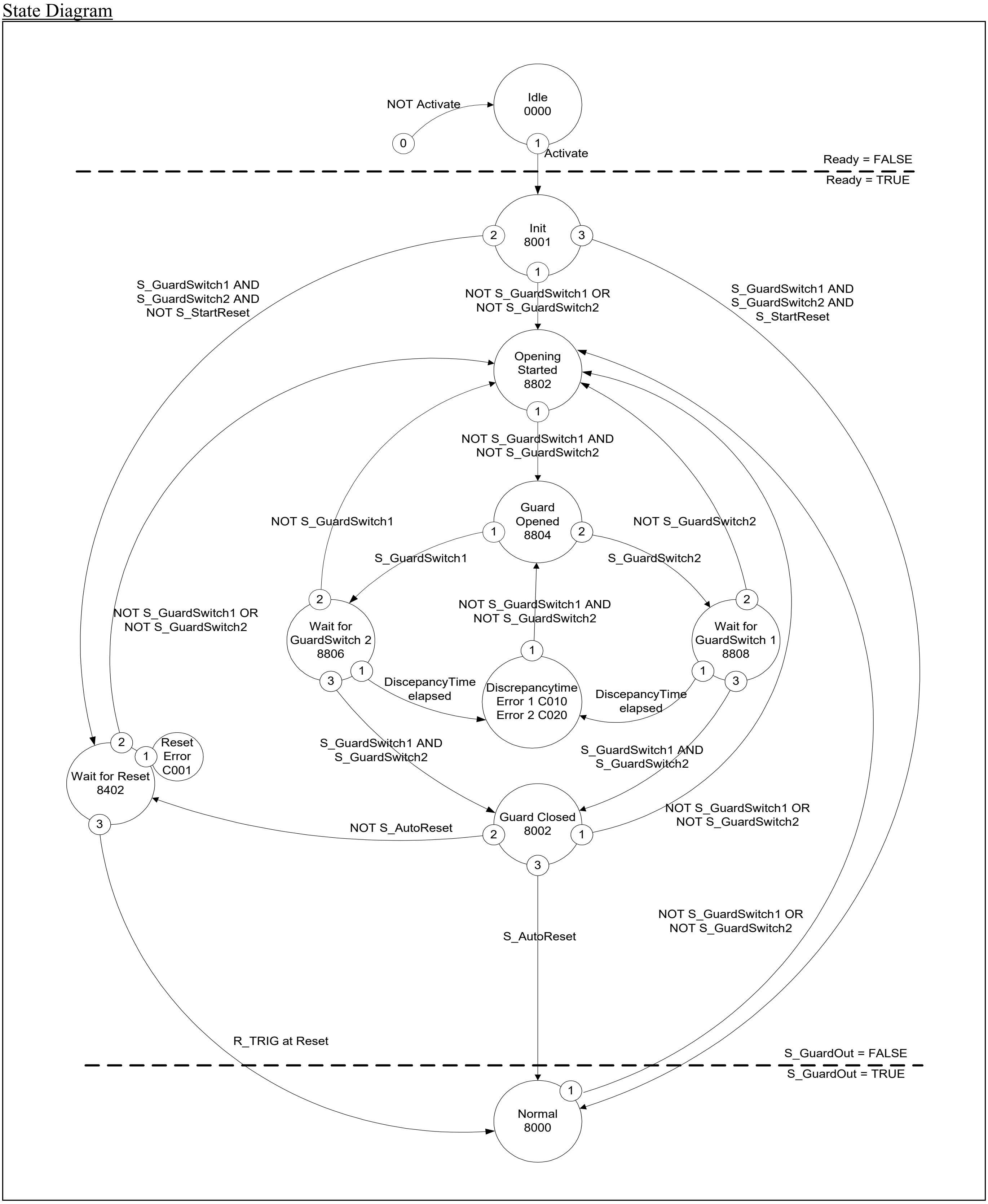
When opening the safety guard, both S_GuardSwitch1 and S_GuardSwitch2 inputs should switch to FALSE. The S_GuardOut output switches to FALSE as soon as one of the switches is set to FALSE. When closing the safety guard, both S_GuardSwitch1 and S_GuardSwitch2 inputs should switch to TRUE.

This FB monitors the symmetry of the switching behavior of both switches. The S_GuardOut output remains FALSE if only one of the contacts has completed an open/close process.

The behavior of the S_GuardOut output depends on the time difference between the switching inputs. The discrepancy time is monitored as soon as the value of both S_GuardSwitch1/S_GuardSwitch2 inputs differs. If the DiscrepancyTime has elapsed, but the inputs still differ, the S_GuardOut output remains FALSE. If the second corresponding S_GuardSwitch1/S_GuardSwitch2 input switches to TRUE within the value specified for the DiscrepancyTime input, the S_GuardOut output is set to TRUE following acknowledgment.

The S_StartReset and S_AutoReset inputs shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

State Diagram



Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Figure 57: State diagram for SF_Guard

Typical Timing Diagrams

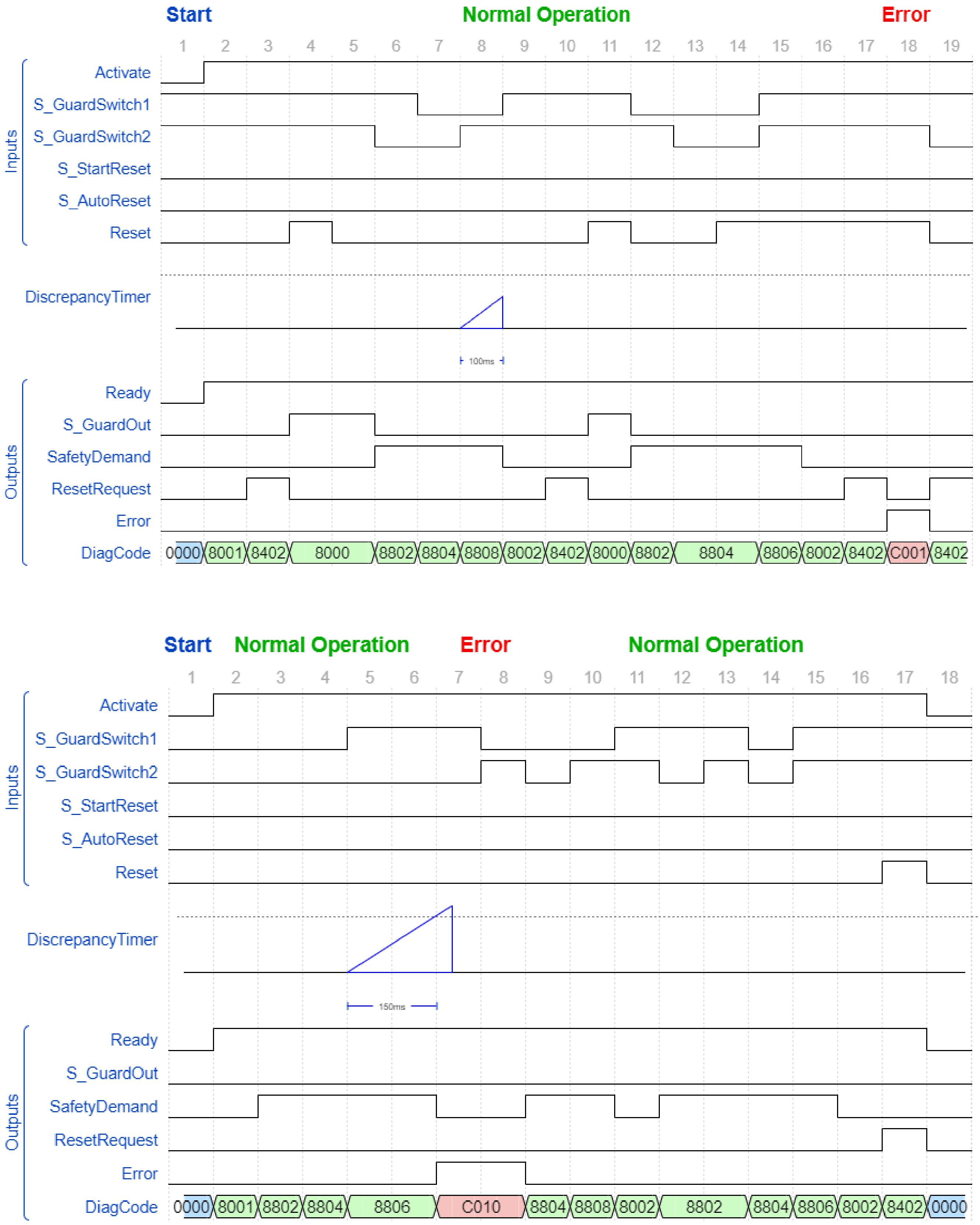


Figure 58: Timing diagrams for SF_Guard

6.16.4 Error Detection

External signals: SAFEBOOL inputs provide inherent error detection. Mechanical setup combines that of an opening and closing switch according to EN 954 (safety guard with two switches). Discrepancy time monitoring for time lag between both mechanical switches' reaction, according to EN 954 (to be considered as "application error" detection, i.e., generated by the application).
An error is detected if the time lag between the first S_GuardSwitch1/S_GuardSwitch2 input and the second is greater than the value for the DiscrepancyTime input. The Error output is set to TRUE.
The function block detects a static TRUE signal at the RESET input.

6.16.5 Error Behavior

The S_GuardOut output is set to FALSE. If the two S_GuardSwitch1 and S_Guardswitch2 inputs are bridged, no error is detected. To leave the Reset error state, the Reset input must be set to FALSE. To leave the discrepancy time errors, the inputs S_GuardSwitch1 and 2 must both be set to FALSE.

6.16.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C001	Reset Error	Static reset detected in state 8402.
		Ready = TRUE
		S_GuardOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C010	Discrepancytime Error 1	DiscrepancyTime elapsed in state 8806.
		Ready = TRUE
		S_GuardOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C020	Discrepancytime Error 2	DiscrepancyTime elapsed in state 8808.
		Ready = TRUE
		S_GuardOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_GuardOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8000	Normal	Safety guard closed and Safe state acknowledged.
		Ready = TRUE
		S_GuardOut = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8001	Init	Function block has been activated.
		Ready = TRUE
		S_GuardOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8802	Opening Started	Complete switching sequence required.
		Ready = TRUE
		S_GuardOut = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8402	Wait for Reset	Waiting for rising trigger at Reset.
		Ready = TRUE
		S_GuardOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

8804	Guard Opened	Guard completely opened.
		Ready = TRUE
		S_GuardOut = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8806	Wait for GuardSwitch2	S_GuardSwitch1 has been switched to TRUE - waiting for S_GuardSwitch2; discrepancy timer started.
		Ready = TRUE
		S_GuardOut = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8808	Wait for GuardSwitch1	S_GuardSwitch2 has been switched to TRUE - waiting for S_GuardSwitch1; discrepancy timer started.
		Ready = TRUE
		S_GuardOut = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8002	Guard Closed	Guard closed. Waiting for Reset, if S_AutoReset = FALSE.
		Ready = TRUE
		S_GuardOut = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

6.17 Safety Guard Interlocking with Locking (Version 2)

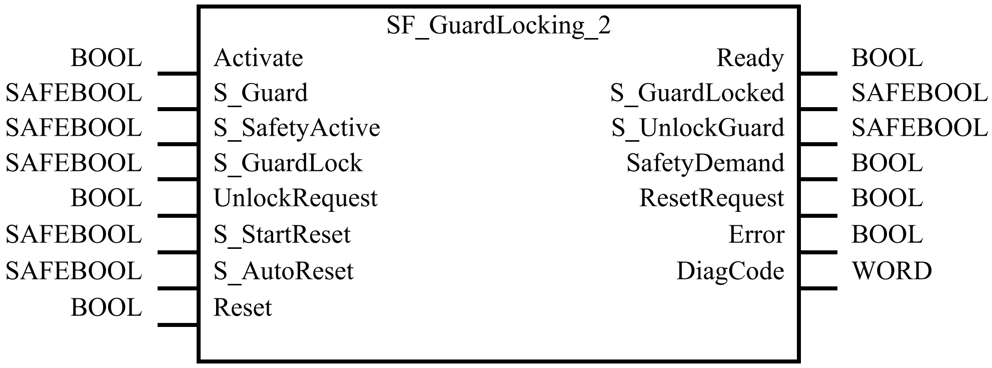
6.17.1 Applicable Safety Standards

Standards	**Requirements**
EN ISO 14120: 2015	3.5 interlocking guard – the hazardous machine functions “covered” by the guard cannot operate until the guard is closed and locked. - if the guard is opened while hazardous machine functions are operating, a stop command is given. - when the guard is closed, the hazardous machine functions “covered” by the guard can operate (the closure of the guard does not, by itself, start the hazardous machine functions) 3.5.1 interlocking guard with a start function control guard special form of interlocking guard which, once it has reached its closed position, gives a command to initiate the hazardous machine function(s) without the use of a separate start control. 3.5.2 interlocking guard with guard locking guard associated with an interlocking device and a guard locking device so that, together with the control system of the machine, the following functions are performed: … - the guard remains closed and locked until the risk due to the hazardous machine functions “covered” by the guard has disappeared;
ISO 14119: 2013	Interlocking devices associated with guards – principles for design and selection
EN ISO 13849-1:2015	5.2.2 Manual reset function. <Note: a positive edge evaluation has the same quality as a negative edge evaluation>
ISO 12100: 2010	6.2.11.4 Restart after power interruption If a hazard could be generated, the spontaneous restart of a machine when it is re-energized after power interruption shall be prevented (for example, by use of a self-maintained relay, contactor or valve).

6.17.2 Interface Description

FB Name	**SF_GuardLocking_2**		
This FB controls an entrance to a hazardous area via an interlocking guard with guard locking (“four state interlocking”).			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
Activate	BOOL	FALSE	See 5.1.1 General Input Parameters
S_Guard	SAFEBOOL	FALSE	Variable. Monitors the guard interlocking. Can be connected to the GuardOut of the SF_Guard FB. FALSE: Guard open. TRUE: Guard closed and guarded area safe.
S_SafetyActive	SAFEBOOL	FALSE	Variable. Status of the hazardous area (EDM), e.g., based on speed monitoring or safe time off delay. FALSE: Machine in "non-safe" state. TRUE: Machine in safe state.
S_GuardLock	SAFEBOOL	FALSE	Variable. Status of the mechanical guard locking. FALSE: Guard is not locked. TRUE: Guard is locked.
UnlockRequest	BOOL	FALSE	Variable. Operator intervention – request to unlock the guard. FALSE: No request. TRUE: Request made.
S_StartReset	SAFEBOOL	FALSE	See 5.1.1 General Input Parameters
S_AutoReset	SAFEBOOL	FALSE	See 5.1.1 General Input Parameters

	Reset	BOOL	FALSE	See 5.1.1 General Input Parameters. Also used to request the guard to be locked again. The quality of the signal must conform to a manual reset device.
VAR_OUTPUT				
	Ready	BOOL	FALSE	See 5.1.2 General Output Parameters
	S_GuardLocked	SAFEBOOL	FALSE	Interface to hazardous area which must be stopped. FALSE: No safe state. TRUE: Safe state.
	S_UnlockGuard	SAFEBOOL	FALSE	Signal to unlock the guard. FALSE: Close guard. TRUE: Unlock guard.
	SafetyDemand	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
	ResetRequest	BOOL	FALSE	Optional. See 5.1.2 General Output Parameters
	Error	BOOL	FALSE	See 5.1.2 General Output Parameters
	DiagCode	WORD	16#0000	See 5.1.2 General Output Parameters
Notes: This FB replaces the original FB SF_GuardLocking of Version 1.0 with changes in the state diagram and additional optional outputs SafetyDemand and ResetRequest.				



6.17.3 Functional Description

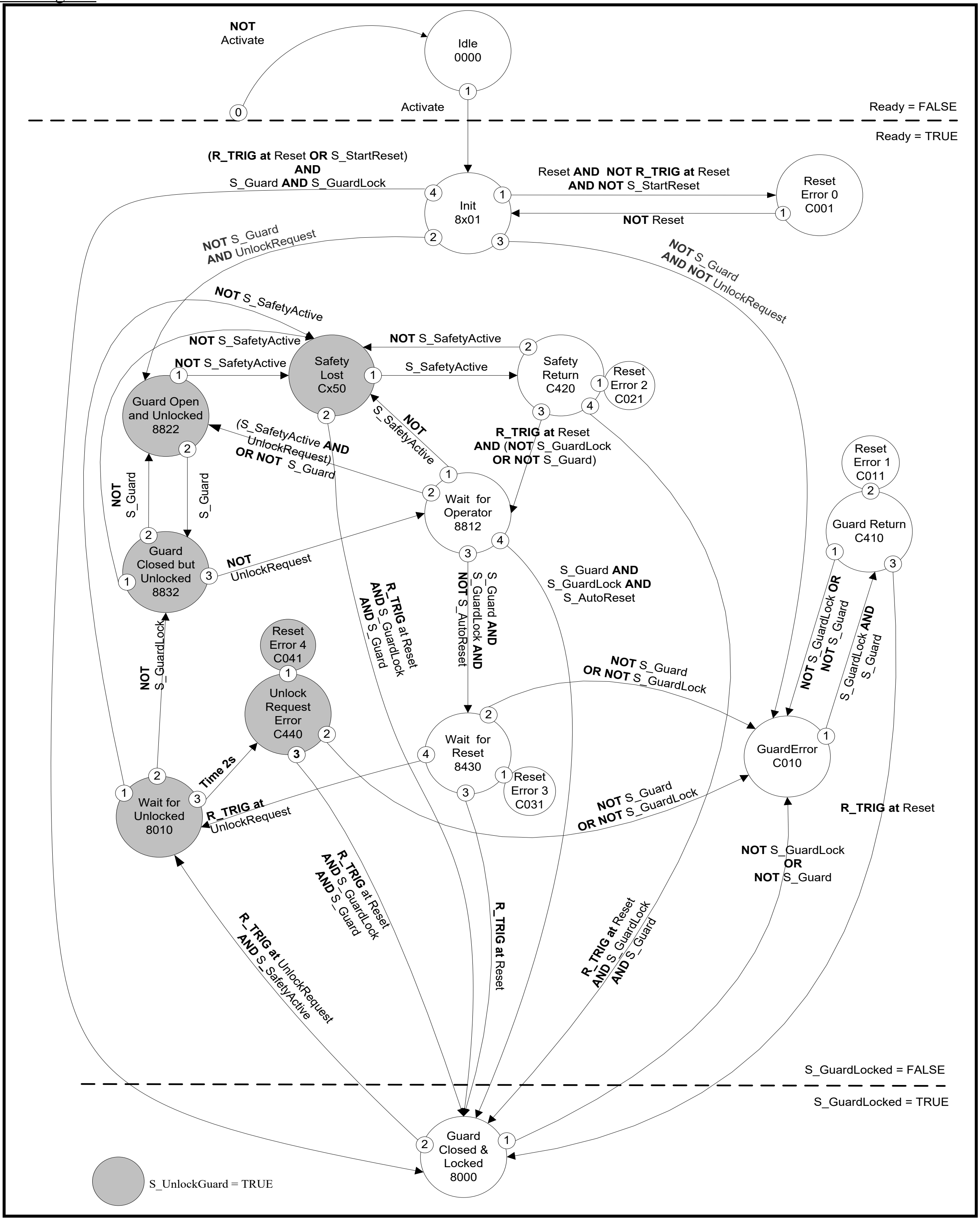
This function controls the guard lock and monitors the position of the guard and the lock. This function block can be used with a mechanical locked switch.
The operator requests to get access to the hazardous area. The guard can only be unlocked when the hazardous area is in a safe state. The guard can be locked if the guard is closed. The machine can be started when the guard is closed and locked. An open guard or unlocked guard will be detected in the event of a safety-critical situation.

The S_StartReset and S_AutoReset inputs shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

Operation Sequence

1.	External	Request to get the hazardous area to a safe state - not part of this FB
2.	In	Feedback from applicable hazardous area that it is in a safe state (via S_SafetyActive)
3.	In	Operator request to unlock the guard (via UnlockRequest)
4.	Out	Enable guard to be opened (via S_UnlockGuard)
5.	In	Guard unlocked (via S_GuardLock). Guard can be opened now. (S_GuardLocked = FALSE)
		Operator opens the guard
6.	In	Monitoring of status guard via S_Guard – signals when guard is closed again
7.	In	Feedback from operator to restart the hazardous area (Reset)
8.	Out	Lock guard guard (S_UnlockGuard)
9.	In	Check if guard is locked (S_GuardLock)
10.	Out	Hazardous area can operate again (S_GuardLocked = TRUE)
11.	Extern	Restart the operation in the hazardous area

State Diagram


NOT Activate
Idle 0000
Activate
Ready = FALSE
Ready = TRUE
(R_TRIG at Reset OR S_StartReset) AND S_Guard AND S_GuardLock
Init 8x01
Reset AND NOT R_TRIG at Reset AND NOT S_StartReset
Reset Error 0 C001
NOT Reset
NOT S_Guard AND UnlockRequest
NOT S_Guard AND NOT UnlockRequest
NOT S_SafetyActive
NOT S_SafetyActive
NOT S_SafetyActive
NOT S_SafetyActive
Safety Lost Cx50
S_SafetyActive
Safety Return C420
Reset Error 2 C021
Guard Open and Unlocked 8822
(S_SafetyActive AND UnlockRequest) OR NOT S_Guard
NOT S_SafetyActive
R_TRIG at Reset AND (NOT S_GuardLock OR NOT S_Guard)
Reset Error 1 C011
Guard Return C410
NOT S_Guard
S_Guard
Wait for Operator 8812
Guard Closed but Unlocked 8832
NOT UnlockRequest
S_Guard AND S_GuardLock AND S_AutoReset
R_TRIG at Reset AND S_GuardLock AND S_Guard
S_Guard AND S_GuardLock AND NOT S_AutoReset
NOT S_GuardLock OR NOT S_Guard
S_GuardLock AND S_Guard
NOT S_GuardLock
Reset Error 4 C041
Unlock Request Error C440
NOT S_Guard OR NOT S_GuardLock
Wait for Reset 8430
Reset Error 3 C031
GuardError C010
Time 2s
R_TRIG at UnlockRequest
Wait for Unlocked 8010
NOT S_Guard OR NOT S_GuardLock
R_TRIG at Reset
NOT S_GuardLock OR NOT S_Guard
R_TRIG at Reset AND S_GuardLock AND S_Guard
R_TRIG at UnlockRequest AND S_SafetyActive
R_TRIG at Reset
R_TRIG at Reset AND S_GuardLock AND S_Guard
S_GuardLocked = FALSE
S_GuardLocked = TRUE
Guard Closed & Locked 8000
S_UnlockGuard = TRUE

Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Figure 59: State diagram for SF_GuardLocking_2

Typical Timing Diagram:

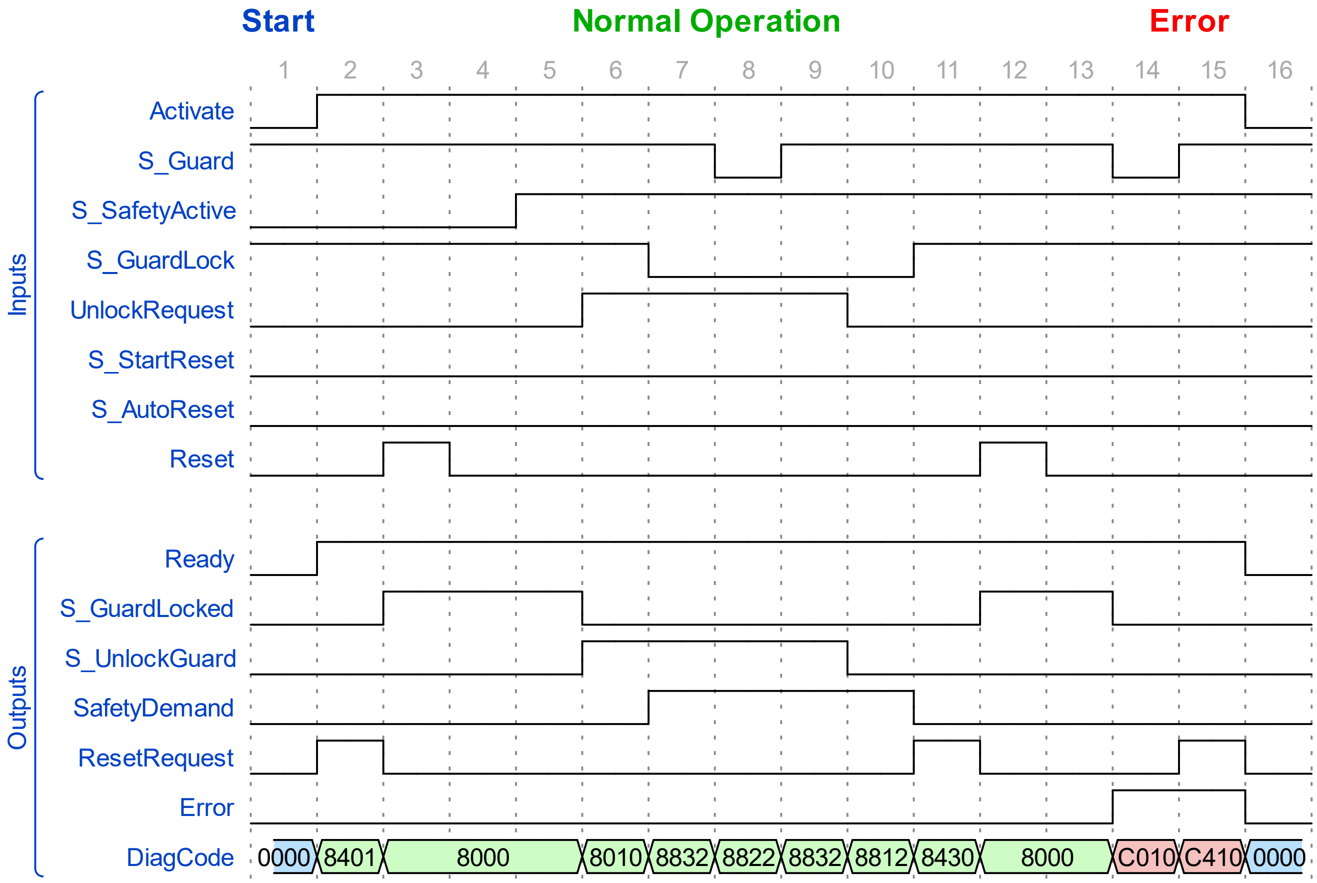


Figure 60: Timing diagram for SF_GuardLocking_2

6.17.4 Error Detection

Static signals are detected at Reset. Errors are detected at the Guard switches.

6.17.5 Error Behavior

In the event of an error the S_GuardLocked and S_UnlockGuard outputs are set to FALSE, the DiagCode output indicates the relevant error code, and the Error output is set to TRUE.
An error must be acknowledged by a rising trigger at the Reset input.

6.17.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C001	Reset Error 0	Static Reset detected in state 8x01.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

C010	Guard Error	S_GuardLock and S_Guard are not TRUE although the door was not requested to be opened.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C011	Reset Error 1	Static Reset detected in state C410.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C410	Guard Return	S_GuardLock and S_Guard become TRUE again after being lost (C010)
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE

Cx50	Safety Lost	Lost safety acknowledge signal
		IF S_Guard = TRUE AND S_GuardLock = TRUE
		THEN x = 4 ELSE x = 0
		
		Output signals for x = 4 (C450):
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE
		
		Output signals for x = 0 (C050):
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C021	Reset Error 2	Static Reset detected in state C420.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

C420	Safety Return	Safety acknowledge signal becomes TRUE again after being lost (Cx50).
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE

C031	Reset Error 3	Static Reset detected in state 8433.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C440	Unlock Request Error	Waiting time to Unlock exceeded.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE

C041	Reset Error 4	Static Reset detected in state C440.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		
		Ready = FALSE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8000	Guard Closed and Locked	Guard is closed and locked.
		
		Ready = TRUE
		S_GuardLocked = TRUE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8x01	Init	Function block was activated and initiated.
		IF S_Guard = TRUE AND S_GuardLock = TRUE
		THEN x = 4 ELSE x = 8
		
		Output signals for x = 4 (8401):
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE
		
		Output signals for x = 8 (8801):
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8430	Wait for Reset	Door is closed and locked, now waiting for operator reset.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

8812	Wait for Operator	Waiting for operator to request to open the door (unlock request).
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8822	Guard Open and Unlocked	Lock is released and guard is open.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8832	Guard Closed but Unlocked	Lock is released but guard is closed.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8010	Wait for Unlocked	S_UnlockGuard is TRUE, however the acknowledge signal S_GuardLock is still TRUE (so waiting for acknowledge <FALSE>)
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

6.18 Safety Guard Interlocking with Locking for switches with serial contacts

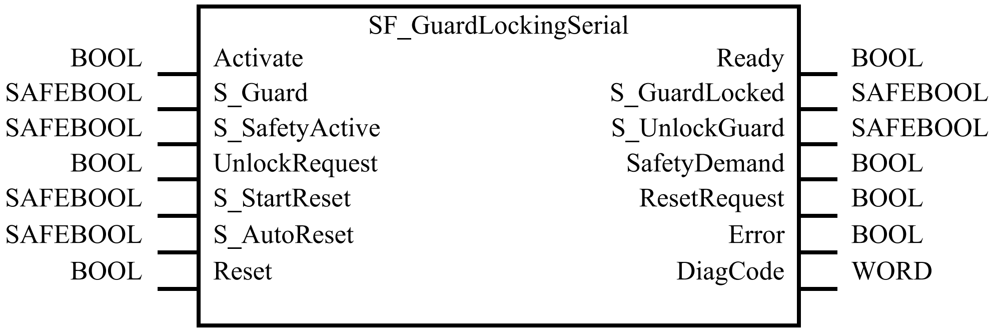
6.18.1 Applicable Safety Standards

Standards	Requirements
ISO 14120: 2015	3.5 interlocking guard – the hazardous machine functions “covered” by the guard cannot operate until the guard is closed and locked. - if the guard is opened while hazardous machine functions are operating, a stop command is given. - when the guard is closed, the hazardous machine functions “covered” by the guard can operate (the closure of the guard does not, by itself, start the hazardous machine functions) 3.5.1 interlocking guard with a start function control guard special form of interlocking guard which, once it has reached its closed position, gives a command to initiate the hazardous machine function(s) without the use of a separate start control. 3.5.2 interlocking guard with guard locking guard associated with an interlocking device and a guard locking device so that, together with the control system of the machine, the following functions are performed: … - the guard remains closed and locked until the risk due to the hazardous machine functions “covered” by the guard has disappeared;
ISO 14119: 2013	Interlocking devices associated with guards – principles for design and selection
EN ISO 13849-1:2015	5.2.2 Manual reset function. <Note: a positive edge evaluation has the same quality as a negative edge evaluation>
ISO 12100-2: 2010	6.2.11.4 Restart after power interruption If a hazard could be generated, the spontaneous restart of a machine when it is re-energized after power interruption shall be prevented (for example, by use of a self-maintained relay, contactor or valve).

6.18.2 Interface Description

FB Name	**SF_GuardLockingSerial**		
This FB controls an entrance to a hazardous area via an interlocking guard with guard locking (“four state interlocking”). The used switch does not distinguish between if the safety door is unlocked but not opened or unlocked and opened. Therefore, we only have the S_Guard input compared to SF_GuardLocking_2.			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
Activate	BOOL	FALSE	See 5.1.1 General Input Parameters
S_Guard	SAFEBOOL	FALSE	Variable. Monitors the guard interlocking. This can be connected to S_GuardOut from the SF_Guard FB FALSE: Guard open. TRUE: Guard closed and guarded area safe.
S_SafetyActive	SAFEBOOL	FALSE	Variable. Status of the hazardous area (EDM), e.g., based on speed monitoring or safe time off delay. FALSE: Machine in "non-safe" state. TRUE: Machine in safe state.
UnlockRequest	BOOL	FALSE	Variable. Operator intervention – request to unlock the guard. FALSE: No request. TRUE: Request made.
S_StartReset	SAFEBOOL	FALSE	See 5.1.1 General Input Parameters
S_AutoReset	SAFEBOOL	FALSE	See 5.1.1 General Input Parameters
Reset	BOOL	FALSE	See 5.1.1 General Input Parameters. General Input Parameters. Also used to request the guard to be locked again. The quality of the signal must conform to a manual reset device.

VAR_OUTPUT			
Ready	BOOL	FALSE	See 5.1.2 General Output Parameters
S_GuardLocked	SAFEBOOL	FALSE	Interface to hazardous area which must be stopped. FALSE: No safe state. TRUE: Safe state.
S_UnlockGuard	SAFEBOOL	FALSE	Signal to unlock the guard. FALSE: Close guard. TRUE: Unlock guard.
SafetyDemand	BOOL	FALSE	See 5.1.2 General Output Parameters
ResetRequest	BOOL	FALSE	See 5.1.2 General Output Parameters
Error	BOOL	FALSE	See 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See 5.1.2 General Output Parameters
Notes: --			



6.18.3 Functional Description

This function controls the guard lock and monitors the position of the combination of guard and lock. This function block can be used with a mechanical locked switch.

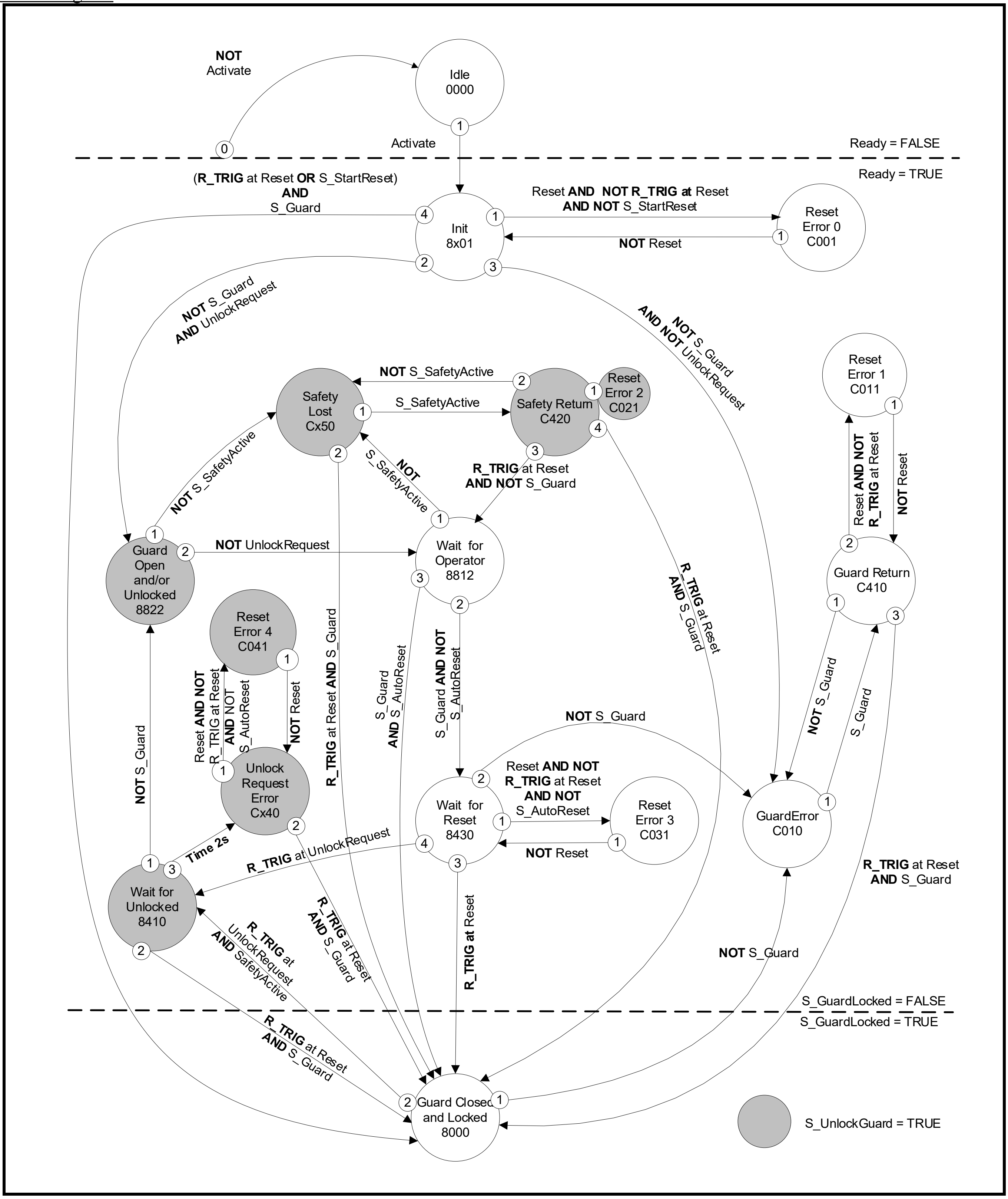
The operator requests to get access to the hazardous area. The guard can only be unlocked when the hazardous area is in a safe state. The guard can be locked if the guard is closed. The machine can be started when the guard is closed and locked. An unlocked guard will be detected to initiate a safety reaction.

The S_StartReset and S_AutoReset inputs shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

Operation Sequence

1.	External	Request to get the hazardous area to a safe state - not part of this FB
2.	In	Feedback from applicable hazardous area that it is in a safe state (via S_SafetyActive)
3.	In	Operator request to unlock the guard (via UnlockRequest)
4.	Out	Enable guard to be opened (via S_UnlockGuard)
5.	In	Guard unlocked (via S_Monitoring). Guard can be opened now. (S_GuardLocked = FALSE)
		Operator opens the guard
6.	In	Feedback from operator to restart the hazardous area (Reset)
7.	Out	Lock guard (S_UnlockGuard)
8.	In	Check if guard is locked (S_Monitoring)
9.	Out	Hazardous area can operate again (S_GuardLocked = TRUE)
10.	Extern	Restart the operation in the hazardous area

State Diagram



Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Figure 61: State diagram for SF_GuardLockingSerial

Typical Timing Diagram

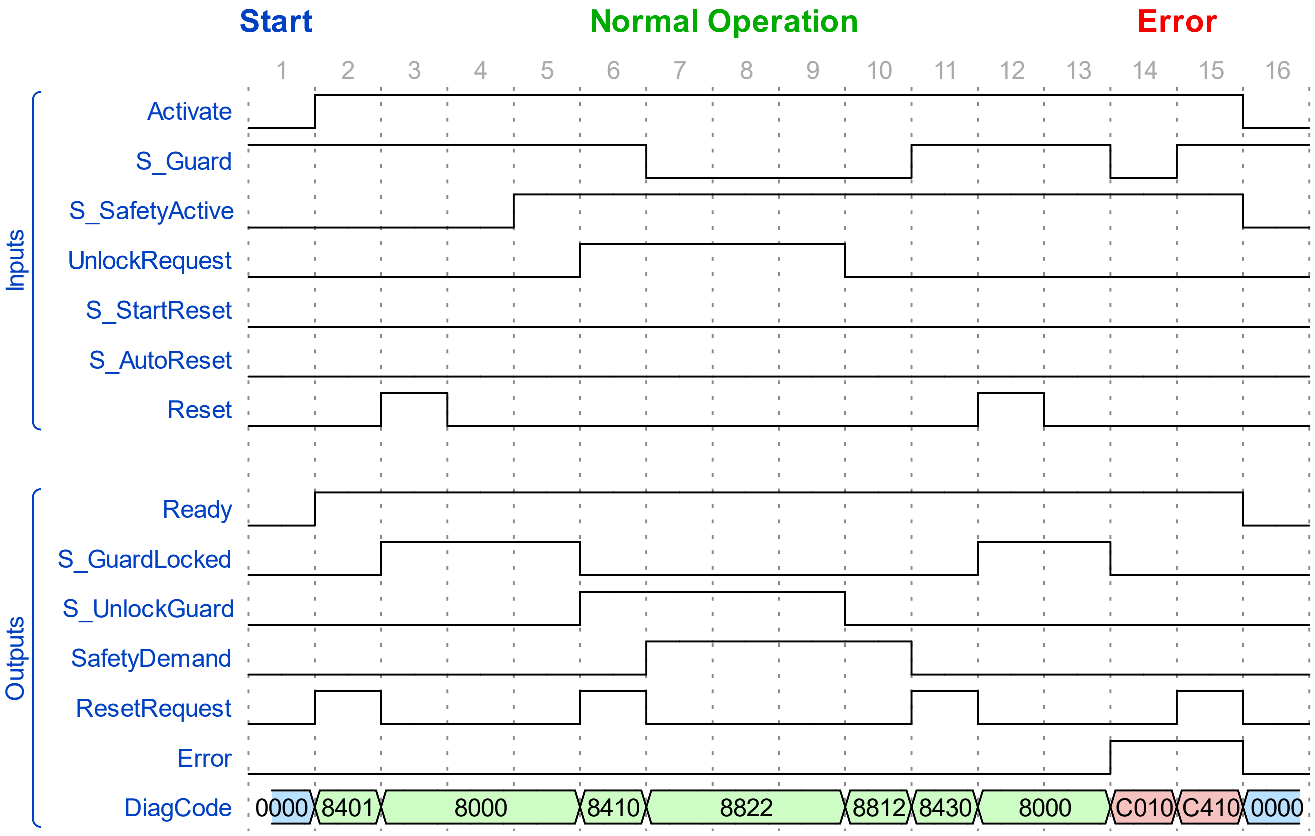


Figure 62: Timing diagram for SF_GuardLockingSerial

6.18.4 Error Detection

Static signals are detected at Reset. Errors are detected at the Guard switches.

6.18.5 Error Behavior

In the event of an error the S_GuardLocked and S_UnlockGuard outputs are set to FALSE, the DiagCode output indicates the relevant error code, and the Error output is set to TRUE.
An error must be acknowledged by a rising trigger at the Reset input.

6.18.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C001	Reset Error 0	Static Reset detected in state 8x01.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C010	Guard Error	S_Guard is not TRUE although the door was not requested to be opened.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

C011	Reset Error 1	Static Reset detected in state C410.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C410	Guard Return	S_Guard becomes TRUE again after being lost (C010).
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE

Cx50	Safety Lost	Lost safety acknowledge signal
		IF S_Guard = TRUE THEN x = 4 ELSE x = 0
		
		Output signals for x = 4 (C450):
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE
		
		Output signals for x = 0 (C050):
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C021	Reset Error 2	Static Reset detected in state C420.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C420	Safety Return	Safety acknowledge signal becomes TRUE again after being lost (Cx50).
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE

C031	Reset Error 3	Static Reset detected in state 8430.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

Cx40	Unlock Request Error	Waiting time to Unlock exceeded.
		IF S_Guard = TRUE THEN x = 4 ELSE x = 0
		Output signals for x = 4 (C440):
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE
		Output signals for x = 0 (C040):
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C041	Reset Error 4	Static Reset detected in state Cx40.
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8000	Guard Closed and Locked	Guard is closed and locked.
		Ready = TRUE
		S_GuardLocked = TRUE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8x01	Init	Function block was activated and initiated.
		
		IF S_Guard = TRUE THEN x = 4 ELSE x = 8
		
		Output signals for x = 4 (8401):
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE
		
		Output signals for x = 8 (8801):
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8430	Wait for Reset	Door is closed and locked, now waiting for operator reset.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

8812	Wait for Operator	Waiting for operator to request to open the door (unlock request).
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8822	Guard Open and/or Unlocked	Guard is unlocked. Door can be closed or open.
		
		Ready = TRUE
		S_GuardLocked = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8410	Wait for Unlocked	S_UnlockGuard is TRUE, however the acknowledge signal S_Guard-Locked is still TRUE (so waiting for acknowledge <FALSE>)
		
		Ready = TRUE
		S_GuardLock = FALSE
		S_UnlockGuard = TRUE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

6.19 Override

6.19.1 Applicable Safety standards

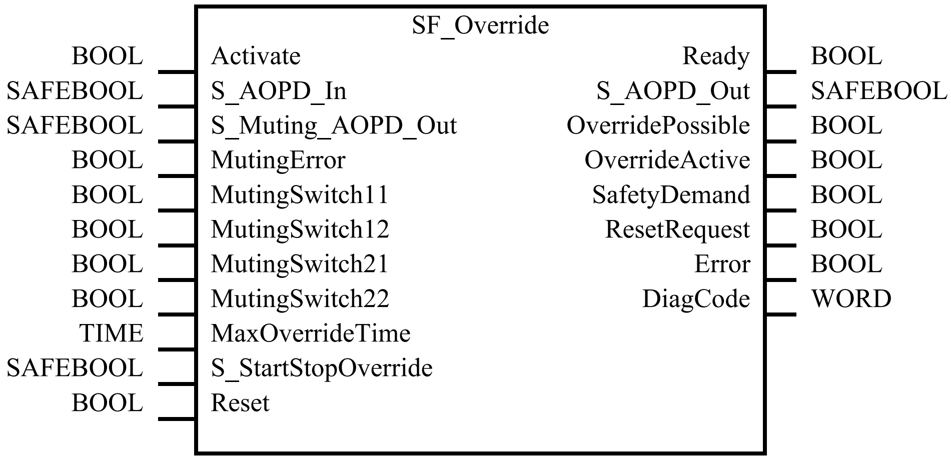
Standards	**Requirements**
EN IEC 62046:2008	5.5.4 Mute dependent override A manually operated, mute dependent override function can be necessary to allow blockages to be removed from the detection zone of the protective equipment. When a mute dependent override function is active, access to the hazardous zone can be possible without actuating the trip function. Mute dependent override shall permit operation of the hazardous elements only in reduced risk conditions. For details of reduced risk conditions see ISO 12100-2, 4.11.9. When a product or transport unit is stopped in the detection zone of the ESPE or of the muting sensors, the muting function shall be cancelled, and all dangerous action once safe operation conditions have been re-established. The override function shall be enabled only when the output of the ESPE is in the OFF-state and/or at least one muting sensor is actuated. From a lockout condition (when a dangerous fault is detected) it shall not be possible to actuate the override function. The mute dependent override function shall: • be activated either: - using a spring return hold-to-run device located so that is not possible to enter the hazardous zone whilst maintaining the action on the hold-to-run device, and so that the hazardous zone is visible while actuating the device. - or using a key operated switch or equally secure momentary action pushbutton when: - the override function is automatically terminated after a correct muting signal sequence is identified, and - no access to the hazardous zone is possible during the override sequence. - an emergency stop can be initiated from the same position. • only be activated when at least one of the muting sensors is actuated. • automatically terminate when all the muting sensors are de-actuated. • automatically terminate after a pre-determined time limit has expired. • only enable those movements that are necessary to permit blockages to be removed from the detection zone of the protective equipment. Measures shall be provided to prevent activation of the mute dependent override function due to a fault or inadvertent operation of the initiating device.
EN ISO 13849-1:2015	5.2.2 Manual reset function. <Note: a positive edge evaluation has the same quality as a negative edge evaluation>

Standards	**Requirements**
EN ISO 12100-2010	6.2.11.9 Control mode for setting, teaching, process changeover, fault-finding, cleaning or maintenance Where, for setting, teaching, process changeover, fault-finding, cleaning or maintenance of machinery, a guard has to be displaced or removed and/or a protective device has to be disabled, and where it is necessary for the purpose of these operations for the machinery or part of the machinery to be put into operation, the safety of the operator shall be achieved using a specific control mode which simultaneously. a) disables all other control modes, b) permits operation of the hazardous elements only by continuous actuation of an enabling device, a two-hand control device or a hold-to-run control device, c) permits operation of the hazardous elements only in reduced risk conditions (for example, reduced speed, reduced power/force, step-by-step, for example, with a limited movement control device), and d) prevents any operation of hazardous functions by voluntary or involuntary action on the machine's sensors. NOTE For some special machinery other protective measures can be appropriate. This control mode shall be associated with one or more of the following measures: - restriction of access to the danger zone as far as possible. - emergency stop control within immediate reach of the operator. - portable control unit (teach pendant) and/or local controls (allowing sight of the controlled elements). See IEC 60204-1.

6.19.2 Interface description

FB-Name		**SF_Override**		
This FB makes it possible to move a product in the production line even when the muting functionality was aborted due to an error. This FB is only applicable in combination with a muting FB.				
VAR_INPUT				
	Name	*Data type*	*Initial value*	*Description, Parameter values*
	Activate	BOOL	FALSE	See 5.1.1 General Input Parameters
	S_AOPD_In	SAFEBOOL	FALSE	Variable. OSSD signal from AOPD. FALSE: Protection field interrupted. TRUE: Protection field not interrupted.
	S_Muting_AOPD_Out	SAFEBOOL	FALSE	Variable. S_AOPD_Out signal from the previous muting function block. FALSE/ TURE: The Status of the Safety related output S_AOPD_Out from the previous muting function block.
	MutingError	BOOL	FALSE	Error output of the previous connected Muting-FB FALSE: No error TRUE: Error in Muting Process
	MutingSwitch11	BOOL	FALSE	Variable. Status of the Muting sensor signal which is connected at the input MutingSwitch11 of the previous muting function block. FALSE: Muting sensor 11 not actuated. TRUE: Workpiece actuates muting sensor 11.
	MutingSwitch12	BOOL	FALSE	Variable. Status of the Muting sensor signal which is connected at the input MutingSwitch12 of the previous muting function block. FALSE: Muting sensor 12 not actuated. TRUE: Workpiece actuates muting sensor 12.

MutingSwitch21	BOOL	FALSE	Variable. Status of the Muting sensor signal which is connected at the input MutingSwitch21 of the previous muting function block. FALSE: Muting sensor 21 not actuated. TRUE: Workpiece actuates muting sensor 21. It shall be noted that this parameter is not connected if the previous muting function is the SF_MutingPar_2Sensor.
MutingSwitch22	BOOL	FALSE	Variable. Status of the Muting sensor signal which is connected at the input MutingSwitch22 of the previous muting function block. FALSE: Muting sensor 22 not actuated. TRUE: Workpiece actuates muting sensor 22. It shall be noted that this parameter is not connected if the previous muting function is the SF_MutingPar_2Sensor.
MaxOverrideTime	Time	T#0s	Constant 0..10 min; Maximum time for the overall Override process. The time is started when the start conditions for the override process are available. The timer is stopped when all the muting sensors are not muted anymore.
S_StartStopOverride	SAFEBOOL	FALSE	Signal for the start and stop of override functionality. A rising edge is needed to start the override functionality. TRUE: If all override conditions are fulfilled, the override process starts. At this moment also the timer for the MaxOverrideTime starts. FALSE: The override process stops. The timer for the MaxOverrideTime continues till the muting process is finished (transition from 8832 to 8002).
Reset	BOOL	FALSE	See 5.1.1 General Input Parameters
VAR_OUTPUT			
Ready	BOOL	FALSE	See 5.1.2 General Output Parameters
S_AOPD_Out	SAFEBOOL	FALSE	Safety related output indicates status of the muted guard or override signal. FALSE: AOPD protection field interrupted and muting not active, or override is not active. TRUE: AOPD protection field not interrupted or muting active, or override is active.
OverridePossible	BOOL	FALSE	Status signaling that override is possible. FALSE: Override not possible TRUE: Override possible
OverrideActive	BOOL	FALSE	Indicates the status of Override process. FALSE: Override not active. TRUE: Override active.
SafetyDemand	BOOL	FALSE	See 5.1.2 General Output Parameters
ResetRequest	BOOL	FALSE	See 5.1.2 General Output Parameters
Error	BOOL	FALSE	See 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See 5.1.2 General Output Parameters
Notes: -			



6.19.3 Functional Description

A manual operated override function can be required to remove blockades in the safety area which resulted during the muting process. If override is active a stop request of the safety equipment is not effective.

This FB SF_Override is only to be used in combination with a muting FB.
In the application program itself, first the muting FB must be processed and then the override FB.

Notice: The Outputs Error and DiagCode of the preconnected Muting are not transmitted to the Outputs Error and DiagCode of the FB SF_Override

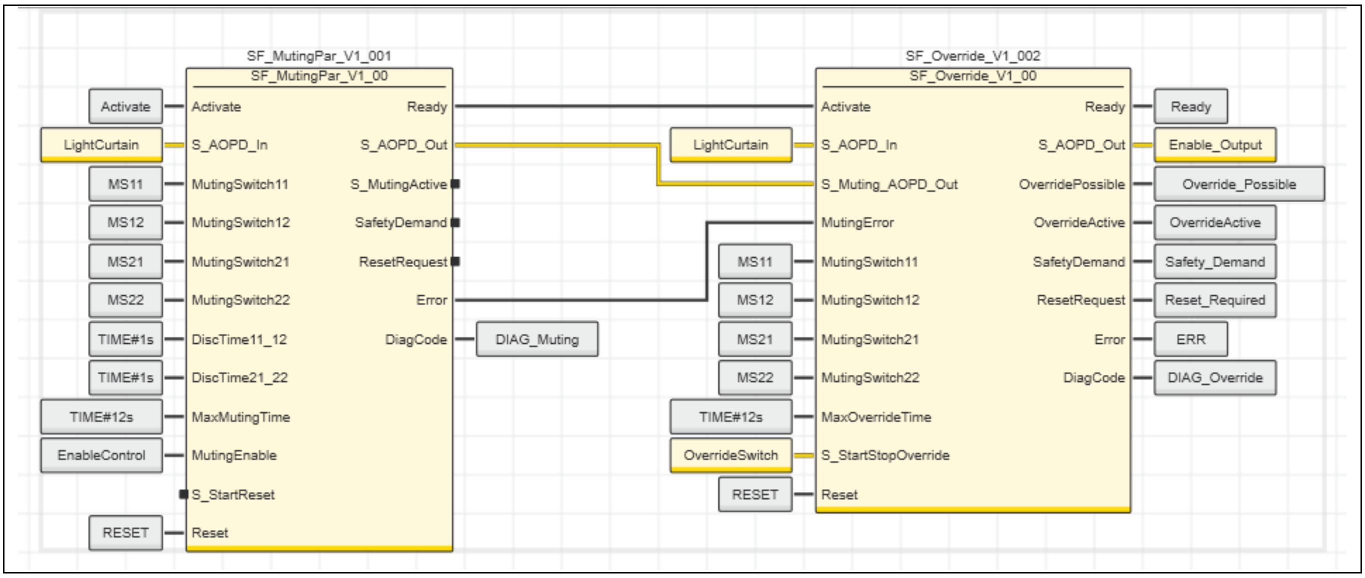


Figure 63: Example Combination of SF_Muting_Par with 4 sensors and SF_Override

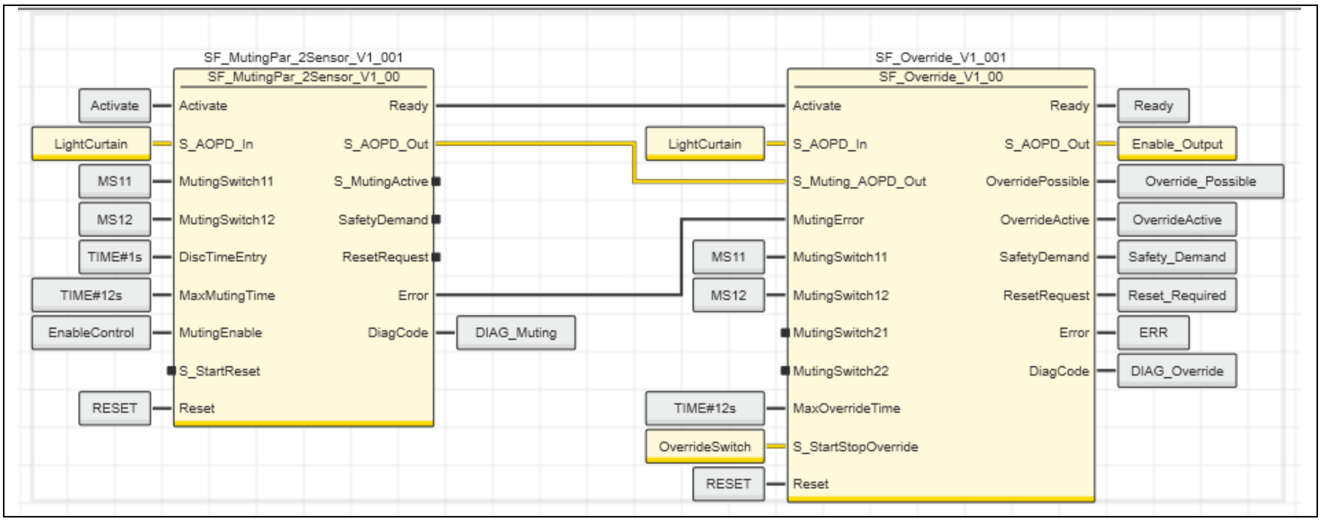


Figure 64: Example Combination of SF_Muting_Par_2Sensor and SF_Override

The override signal (S_AOPD_Out of the SF_Override FB) is set by the FB if:

- the pre-connected muting FB shows an error.
- an applicable S_StartStopOverride signal has a rising edge and a static TRUE.
- the safeguard (e.g., light curtain) is interrupted and/or
- at least one muting sensor is blocked.

The override signal (S_AOPD_Out of the SF_Override FB) is reset by the FB if:

- all muting sensors are ‘clear’ and the safeguard (e.g., light curtain) is not interrupted
- the applicable maximum override time is expired.
- the S_StartStopOverride signal is FALSE.

State diagram

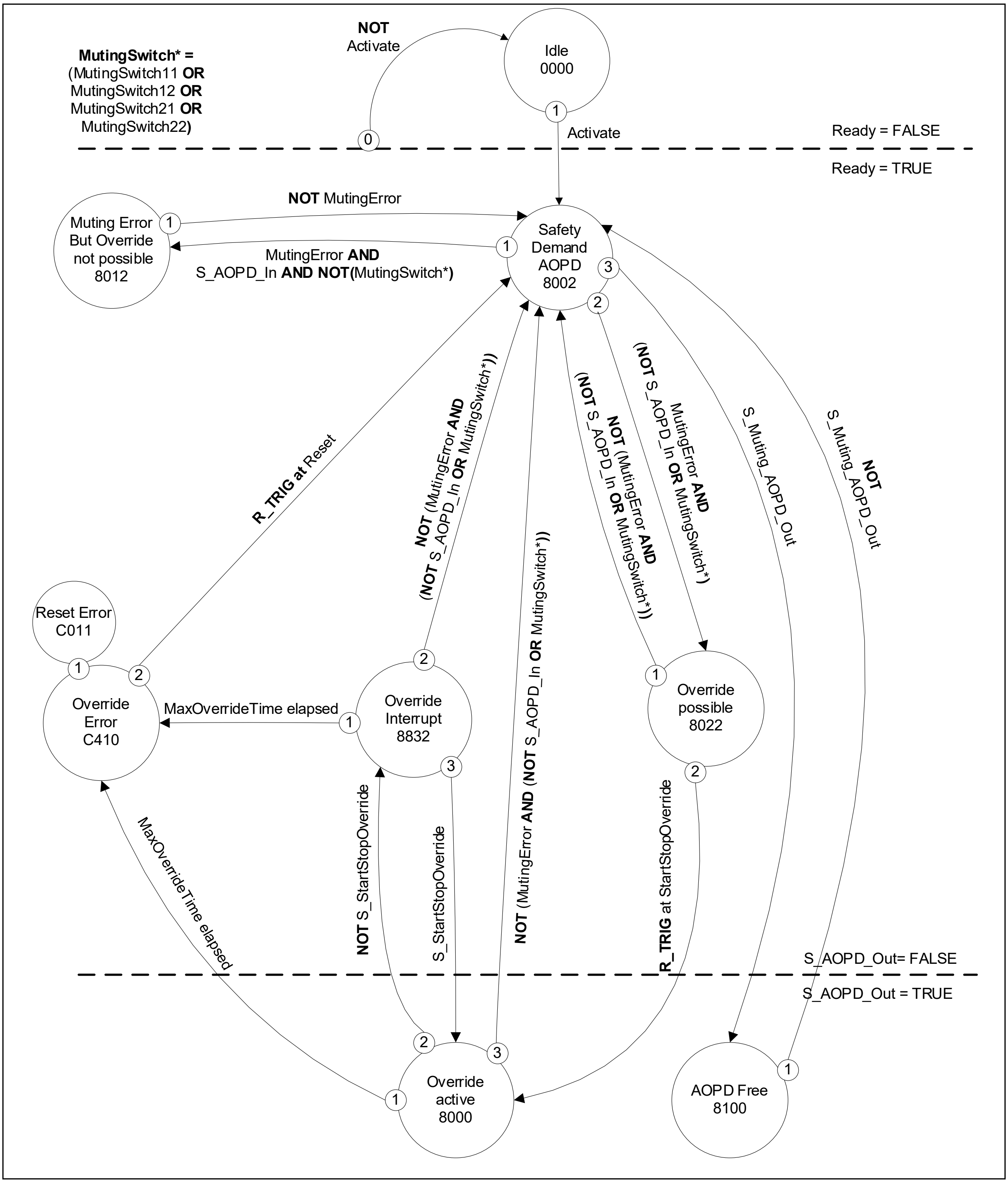

MutingSwitch* = (MutingSwitch11 OR MutingSwitch12 OR MutingSwitch21 OR MutingSwitch22)
NOT Activate
Idle 0000
Activate
Ready = FALSE
Ready = TRUE
NOT MutingError
Muting Error But Override not possible 8012
MutingError AND S_AOPD_In AND NOT(MutingSwitch*)
Safety Demand AOPD 8002
R_TRIG at Reset
NOT (MutingError AND (NOT S_AOPD_In OR MutingSwitch*))
NOT (MutingError AND (NOT S_AOPD_In OR MutingSwitch*))
NOT (MutingError AND (NOT S_AOPD_In OR MutingSwitch*))
MutingError AND (NOT S_AOPD_In OR MutingSwitch*)
S_Muting_AOPD_Out
NOT S_Muting_AOPD_Out
Reset Error C011
Override Error C410
MaxOverrideTime elapsed
Override Interrupt 8832
Override possible 8022
NOT S_StartStopOverride
S_StartStopOverride
R_TRIG at StartStopOverride
MaxOverrideTime elapsed
S_AOPD_Out= FALSE
S_AOPD_Out = TRUE
Override active 8000
AOPD Free 8100

Figure 65: State diagram SF_Override

Typical Timing Diagram

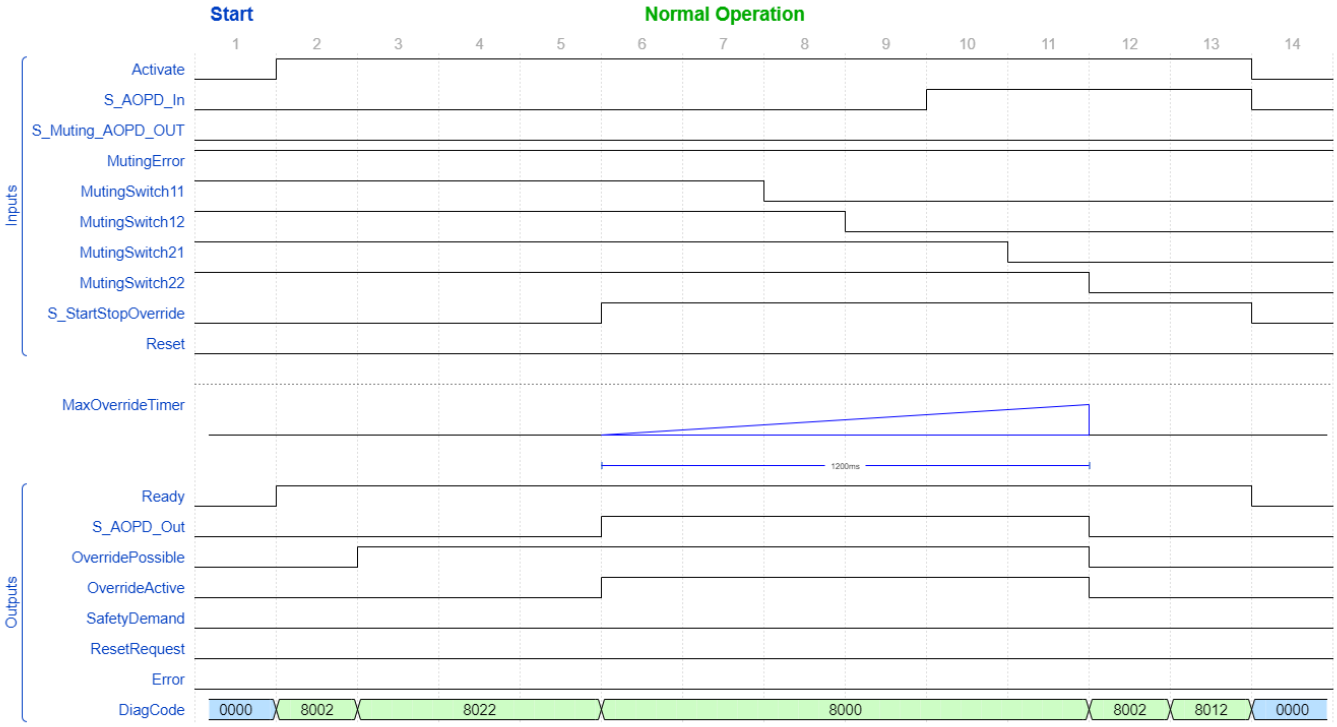


Figure 66 : Timing Diagram of SF_Override with sequential muting

This diagram shows the functionality of the Override FB in combination with sequential muting. This is visible in the transition of the muting inputs while in state 8000. This is related to the moving of the object in the muted area.

6.19.1 Error Detection

Static signals are detected at Reset and MaxOverrideTime elapsed.

6.19.2 Error Behavior

In the event of an error the Error output is set to TRUE, the OverridePossible is de-activated, the OverrideActive is de-activated, and the DiagCode output indicates the relevant error code.
An error must be acknowledged by a rising trigger at the Reset input.

6.19.3 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C011	Reset Error	Static Reset condition detected after FB activation.
		Ready = TRUE
		S_AOPD_Out = FALSE
		OverridePossible = FALSE
		OverrideActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

C410	Override Error	Max Override time elapsed
		Ready = TRUE
		S_AOPD_Out = FALSE
		OverridePossible = FALSE
		OverrideActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		
		Ready = FALSE
		S_AOPD_Out = FALSE
		OverridePossible = FALSE
		OverrideActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8002	Safety Demand AOPD	Protection field interrupted and muting not active, or override is not active and the timer for the MaxOverrideTime will be reset.
		Ready = TRUE
		S_AOPD_Out = FALSE
		OverridePossible = FALSE
		OverrideActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8012	Muting Error but Override not possible	The pre-connected muting FB shows an error but the safeguard (e.g., light curtain) is not interrupted and no muting sensor is blocked.
		
		Ready = TRUE
		S_AOPD_Out = FALSE
		OverridePossible = FALSE
		OverrideActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8022	Override Possible	The pre-connected muting FB shows an error and the safeguard (e.g., light curtain) is interrupted and/or at least one muting sensor is blocked
		Ready = TRUE
		S_AOPD_Out = FALSE
		OverridePossible = TRUE
		OverrideActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8832	Override Interrupt	The override start signal is set to FALSE during override process. The time for the MaxOverrideTime is still running.
		Ready = TRUE
		S_AOPD_Out = FALSE
		OverridePossible = TRUE
		OverrideActive = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8000	Override Active	Override is active and the timer for the MaxOverrideTime is starting to run.
		
		Ready = TRUE
		S_AOPD_Out = TRUE
		OverridePossible = TRUE
		OverrideActive = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8100	AOPD Free	S_AOPD_Out from the pre-connected function block is TRUE.
		Ready = TRUE
		S_AOPD_Out = TRUE
		OverridePossible = FALSE
		OverrideActive = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

7 Safety Function Blocks Post Processing

In this chapter the FBs are listed for the post-processing phase conforming to Figure 6: Layers in the architectural model.

7.1 Safety Request

7.1.1 Applicable Safety Standards

Standards	**Requirements**
IEC 60204-1:2016	9.3.6 Suspension of safety functions and/or protective measures Where it is necessary to suspend safety functions and/or protective measures (for example for setting or maintenance purposes), protection shall be ensured by: - disable all other operating (control) modes. - permit operation only using a hold-to-run device or by a similar control device positioned so as to permit sight of the hazardous elements. - permit operation of the hazardous elements only in reduced risk conditions (e.g., reduced speed, reduced power / force, step-by-step operation, e.g., with a limited movement control device); – prevent any operation of hazardous functions by voluntary or involuntary action on the machine’s sensors.
EN ISO 13849-1:2015	5.2.2 Manual reset function
ISO 12100: 2010	6.2.11.2 Starting of an internal power source/switching on an external power supply. 6.2.11.4 Restart after power interruption

7.1.2 Interface Description

The function block represents the interface between the user program and system environment.

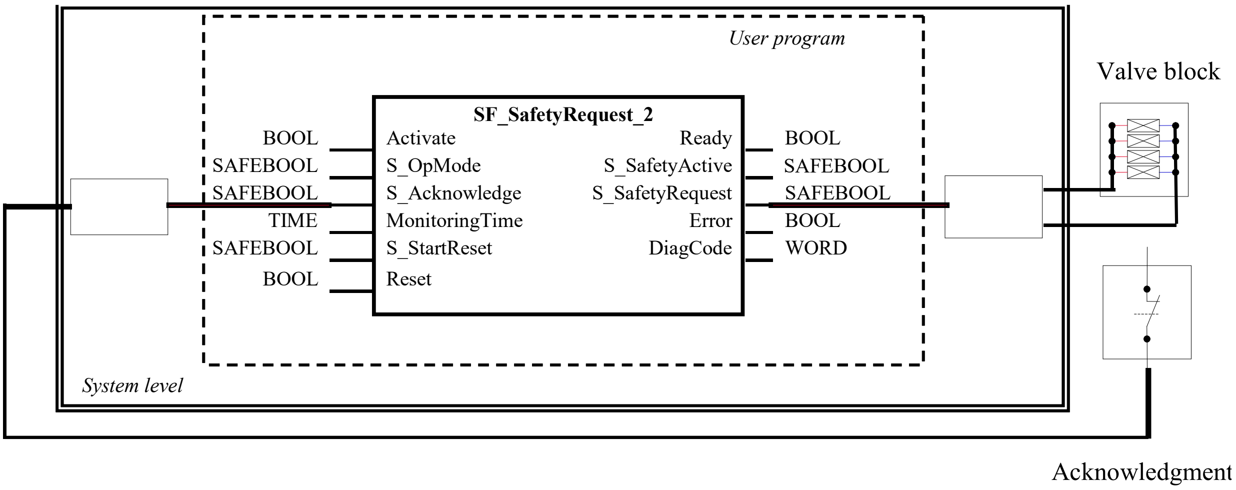
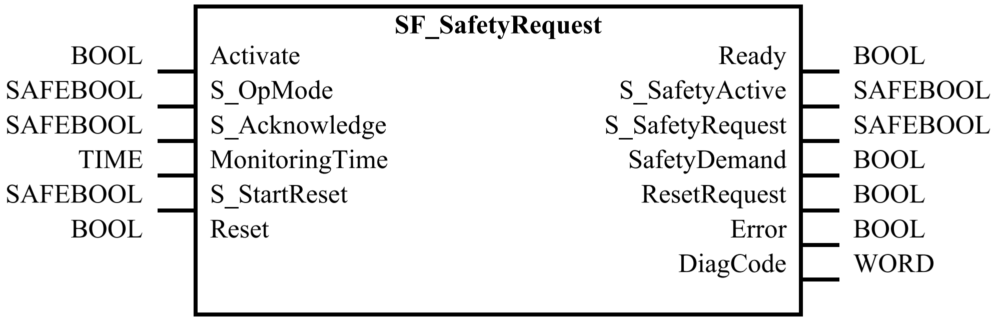


Figure 67: Example SF_SafetyRequest.

FB Name	**SF_SafetyRequest**		
This function block provides the interface to a generic actuator, e.g., a safety drive or safety valve, to place the actuator in a safe state.			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
S_OpMode	SAFEBOOL	FALSE	Variable. Requested mode of a generic safe actuator. FALSE: Safe mode is requested. TRUE: Operation mode is requested.

S_Acknowledge	SAFEBOOL	FALSE	Variable. Confirmation of the generic actuator if actuator is in the Safe state. FALSE: Operation mode (non-safe). TRUE: Safe mode.
MonitoringTime	TIME	T#0s	Constant. Monitoring of the response time between the safety function request (S_OpMode set to FALSE) and the actuator acknowledgment (S_Acknowledge switches to TRUE).
S_StartReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
Reset	BOOL	FALSE	See Section 5.1.1 General Input Parameters with the functionality as an error removed acknowledge
VAR_OUTPUT			
Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
S_SafetyActive	SAFEBOOL	FALSE	Confirmation of the Safe state. FALSE: Non-safe state. TRUE: Safe state.
S_SafetyRequest	SAFEBOOL	FALSE	Request to place the actuator in a safe state. FALSE: Safe state is requested. TRUE: Non-safe state.
SafetyDemand	BOOL	FALSE	See 5.1.2 General Output Parameters
ResetRequest	BOOL	FALSE	See 5.1.2 General Output Parameters
Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters
Notes: --			



7.1.3 Functional Description

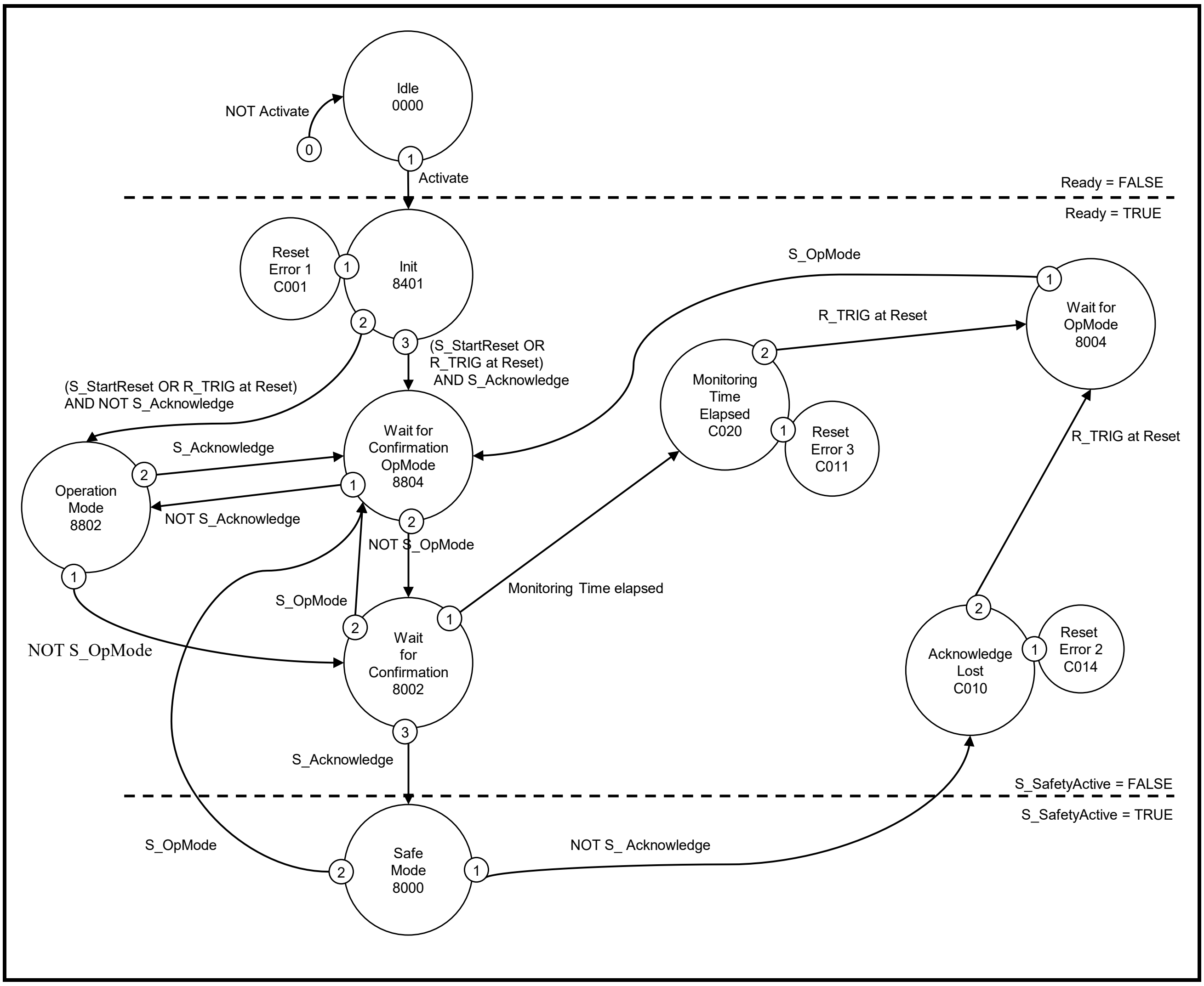
This FB provides the interface between the safety-related system and a generic actuator. This means that the safety-related functions of the actuator are available within the application program. However, there are only two binary signals to control the Safe state of the generic actuator, i.e., one for requesting and one for receiving the confirmation.

The safety function will be provided by the actuator itself. Therefore, the FB only initiates the request, monitors it, and sets the output when the actuator acknowledges the Safe state. This will be indicated with the "S_SafetyActive" output.

This FB does not define any generic actuator-specific parameters. They should have been specified in the generic actuator itself. It switches the generic actuator from the operation mode to a safe state.

The additional input S_StartReset offers the possibility of an automatic cold start as it is defined for the other FBs. Setting the input to FALSE the compatibility to the origin FB is given.

State Diagram



Note 1: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Figure 68: State diagram for SF_SafetyRequest

Typical Timing Diagram

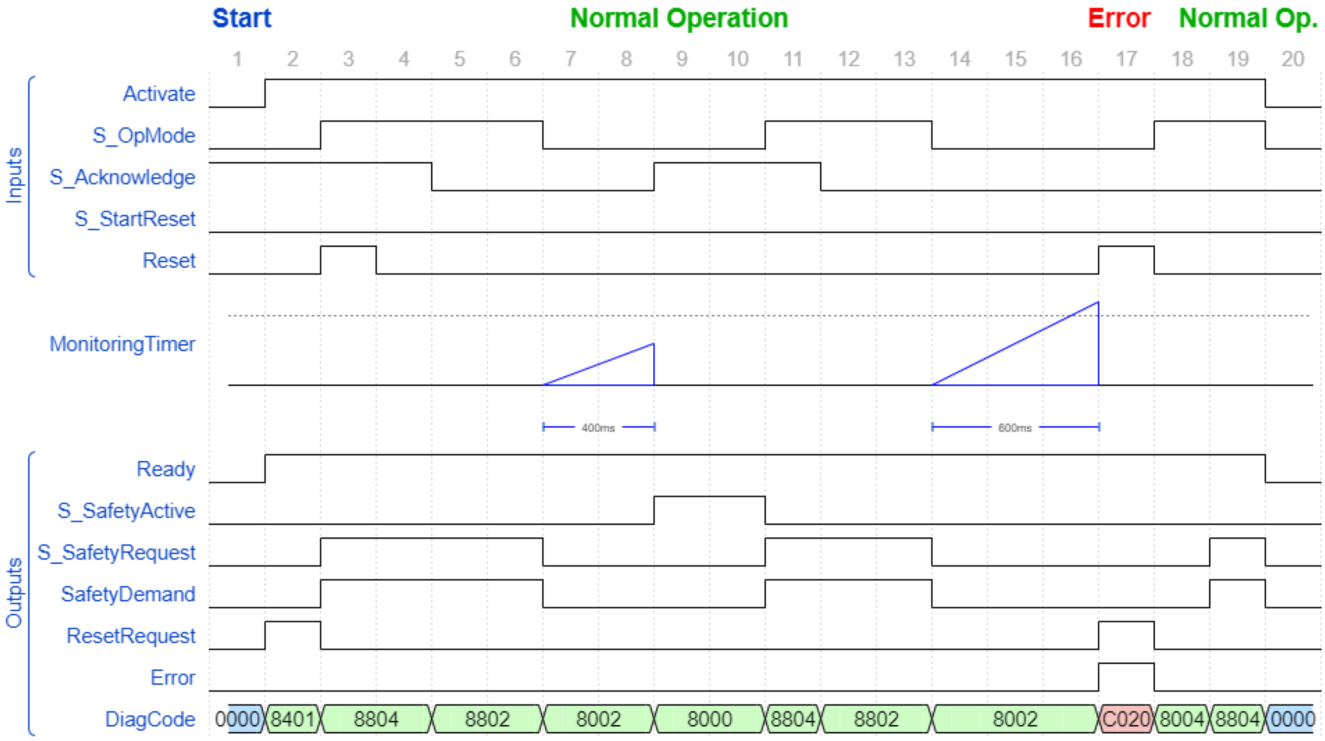


Figure 69: Timing diagram for SF_SafetyRequest

7.1.4 Error Detection

The FB detects whether the actuator does not enter the Safe state within the monitoring time.
The FB detects whether the acknowledge signal is lost while the request is still active.
The FB detects a static Reset signal.

External FB errors:
There are no external errors, since there is no error bits/information provided by the generic actuator.

7.1.5 Error Behavior

In the event of an error, the S_SafetyActive output is set to FALSE.
An error must be acknowledged by a rising trigger at the Reset input. To continue the function block after this reset, the S_OpMode request must be set to TRUE or S_Acknowledge must become TRUE.

7.1.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C010	Acknowledge Lost	Acknowledgment lost while in the Safe state.
		Ready = TRUE
		S_SafetyActive = FALSE
		S_SafetyRequest = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE

C020	MonitoringTime Elapsed	S_OpMode request could not be completed within the monitoring time.
		Ready = TRUE
		S_SafetyActive = FALSE
		S_SafetyRequest = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

C001	Reset Error 1	Static Reset detected in state 8401 Init.
		Ready = TRUE
		S_SafetyActive = FALSE
		S_SafetyRequest = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C014	Reset Error 2	Static Reset detected in state C002 (Acknowledge Lost).
		Ready = TRUE
		S_SafetyActive = FALSE
		S_SafetyRequest = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C011	Reset Error 3	Static Reset detected in state C003 (MonitoringTime Elapsed).
		Ready = TRUE
		S_SafetyActive = FALSE
		S_SafetyRequest = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_SafetyActive = FALSE
		S_SafetyRequest = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8000	Safe Mode	Actuator is in a safe mode.
		Ready = TRUE
		S_SafetyActive = TRUE
		S_SafetyRequest = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8401	Init	State after Activate is set to TRUE or after a rising trigger at Reset.
		Ready = TRUE
		S_SafetyActive = FALSE
		S_SafetyRequest = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

8802	Operation Mode	Operation mode without Acknowledge of safe mode.
		Ready = TRUE
		S_SafetyActive = FALSE
		S_SafetyRequest = TRUE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8804	Wait for Confirmation OpMode	Operation mode with Acknowledge of safe mode.
		Ready = TRUE
		S_SafetyActive = FALSE
		S_SafetyRequest = TRUE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8002	Wait for Confirmation	Waiting for confirmation from the drive (system interface).
		Ready = TRUE
		S_SafetyActive = FALSE
		S_SafetyRequest = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8004	Wait for OpMode	Error was removed. However, S_OpMode must be set to TRUE or S_Acknowledge must become TRUE before the FB can be continued.
		Ready = TRUE
		S_SafetyActive = FALSE
		S_SafetyRequest = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

7.2 OutControl

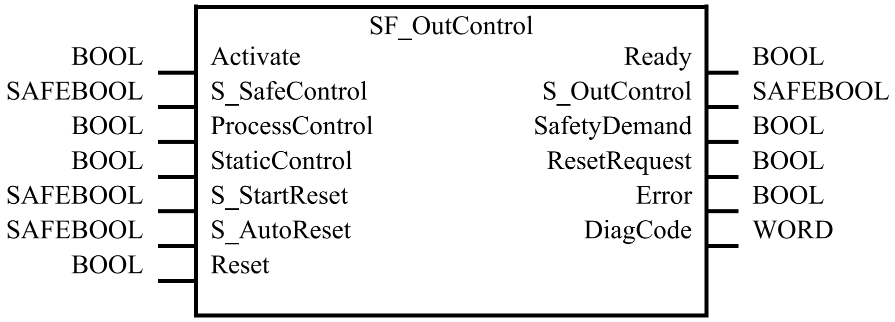
7.2.1 Applicable Safety Standards

Standards	**Requirements**
IEC 60204-1:2009	9.2.2: Stop functions: Stop function categories; Category 0 - stopping by immediate removal of power to the machine actuators (i.e., an uncontrolled stop …) 9.2.5.2: Start: The start of an operation shall be possible only when all the relevant safety functions and/or protective measures are in place and are operational except for conditions as described in 9.2.4. Suitable interlocks shall be provided to secure correct sequential starting.
EN ISO 13849-1:2015	5.2.1 Safety-related stop function A safety-related stop function (e.g., initiated by a safeguard) shall, as soon as necessary after actuation, put the machine in a safe state. Such a stop shall have priority over a stop for operational reasons. 5.2.3 Start/restart function. A restart shall take place automatically only if a hazardous situation cannot exist. 5.2.8 Fluctuations, loss, and restoration of power sources When fluctuations in energy levels outside the design operating range occur, including loss of energy supply, the SRP/CS shall continue to provide or initiate output signal(s) which will enable other parts of the machine system to maintain a safe state.
ISO 12100: 2010	6.2.11.2 Starting of an internal power source/switching on an external power supply 6.2.11.4 Restart after power interruption
EN ISO 13849-1:2015	5.2.2 Manual reset function

7.2.2 Interface Description

FB Name	**SF_OutControl**		
Control of a safety output with a signal from the functional application and a safety signal with optional startup inhibits.			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
S_SafeControl	SAFEBOOL	FALSE	Variable. Control signal of the preceding safety FB. Typical function block signals from the library (e.g., SF_EStop, SF_Guard, SF_TwoHandControlTypeII, and/or others). FALSE: The preceding safety FBs are in safe state. TRUE: The preceding safety FB's enable safety control.
ProcessControl	BOOL	FALSE	Variable or constant. Control signal from the functional application. FALSE: Request to set S_OutControl to FALSE. TRUE: Request to set S_OutControl to TRUE.
StaticControl	BOOL	FALSE	Constant. Optional conditions for process control. FALSE: Dynamic change at ProcessControl (FALSE => TRUE) required after block activation or triggered safety function. Additional function start required. TRUE: No dynamic change at ProcessControl (FALSE => TRUE) required after block activation or triggered safety function.
S_StartReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
S_AutoReset	SAFEBOOL	FALSE	See Section 5.1.1 General Input Parameters
Reset	BOOL	FALSE	See Section 5.1.1 General Input Parameters

VAR_OUTPUT				
	Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
	S_OutControl	SAFEBOOL	FALSE	Controls connected actuators. FALSE: Disable connected actuators. TRUE: Enable connected actuators.
	SafetyDemand	BOOL	FALSE	See 5.1.2 General Output Parameters
	ResetRequest	BOOL	FALSE	See 5.1.2 General Output Parameters
	Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
	DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters
Notes: -				



7.2.3 Functional Description

General:

The SF_OutControl FB is an output driver for a safety output.
The safety output is controlled via S_OutControl using a signal from the functional application (ProcessControl/BOOL to control the process) and a signal from the safety application (S_SafeControl/SAFEBOOL to control the safety function).

Optional conditions for process control (ProcessControl):

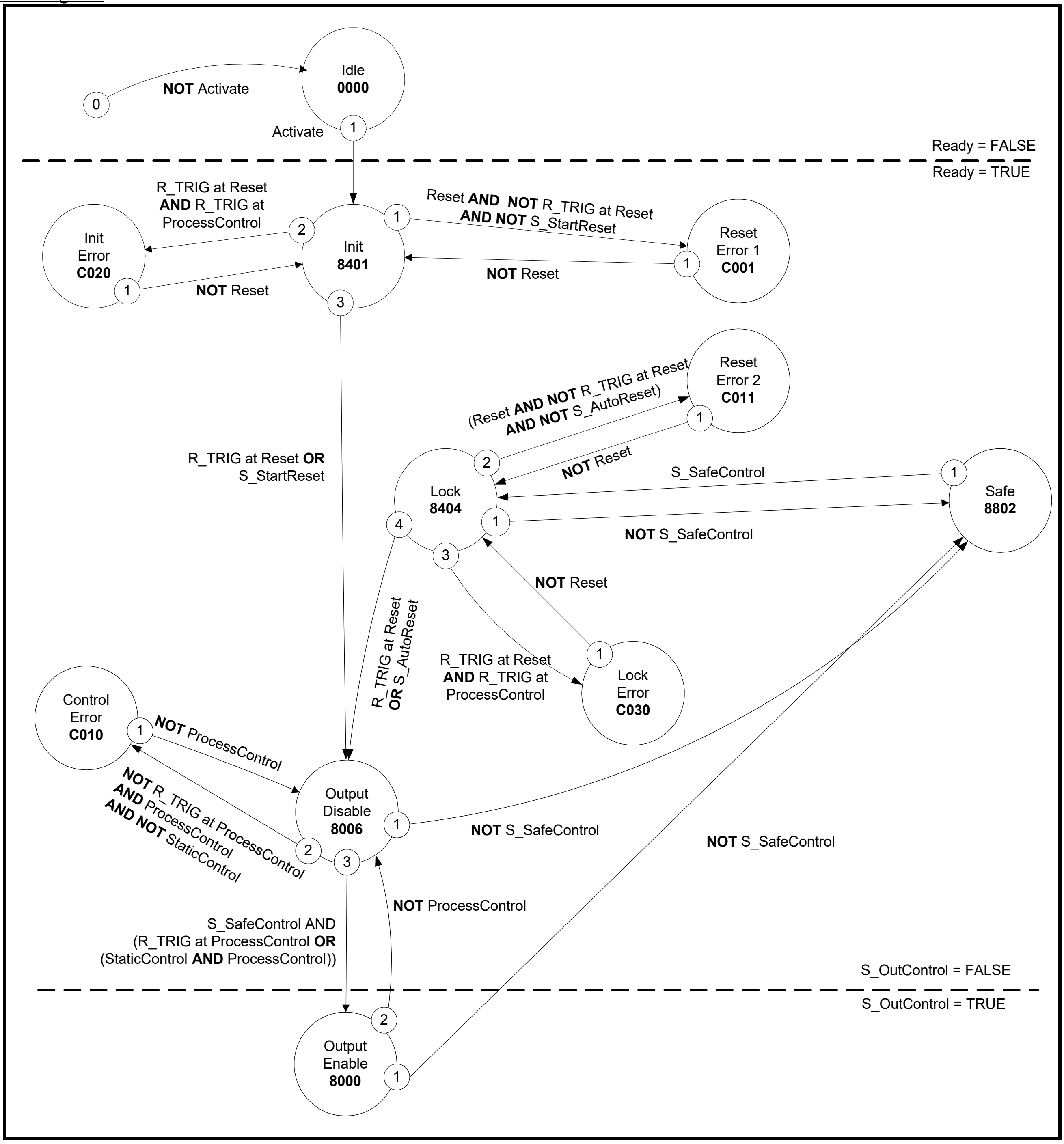
- An additional function start (ProcessControl FALSE => TRUE) is required following block activation or feedback of the safe signal (S_SafeControl). A static TRUE signal at ProcessControl does **not** set S_OutControl to TRUE.
- An additional function start (ProcessControl FALSE => TRUE) is **not** required following block activation or feedback of the safe signal (S_SafeControl). A static TRUE signal at ProcessControl sets S_OutControl to TRUE if the other conditions have been met.

Optional startup inhibits:

- Startup inhibit after function block activation.
- Startup inhibit after interruption of the protective device.

The StaticControl, S_StartReset and S_AutoReset inputs shall only be activated if it is ensured that no hazardous situation can occur when the PES is started.

State Diagram

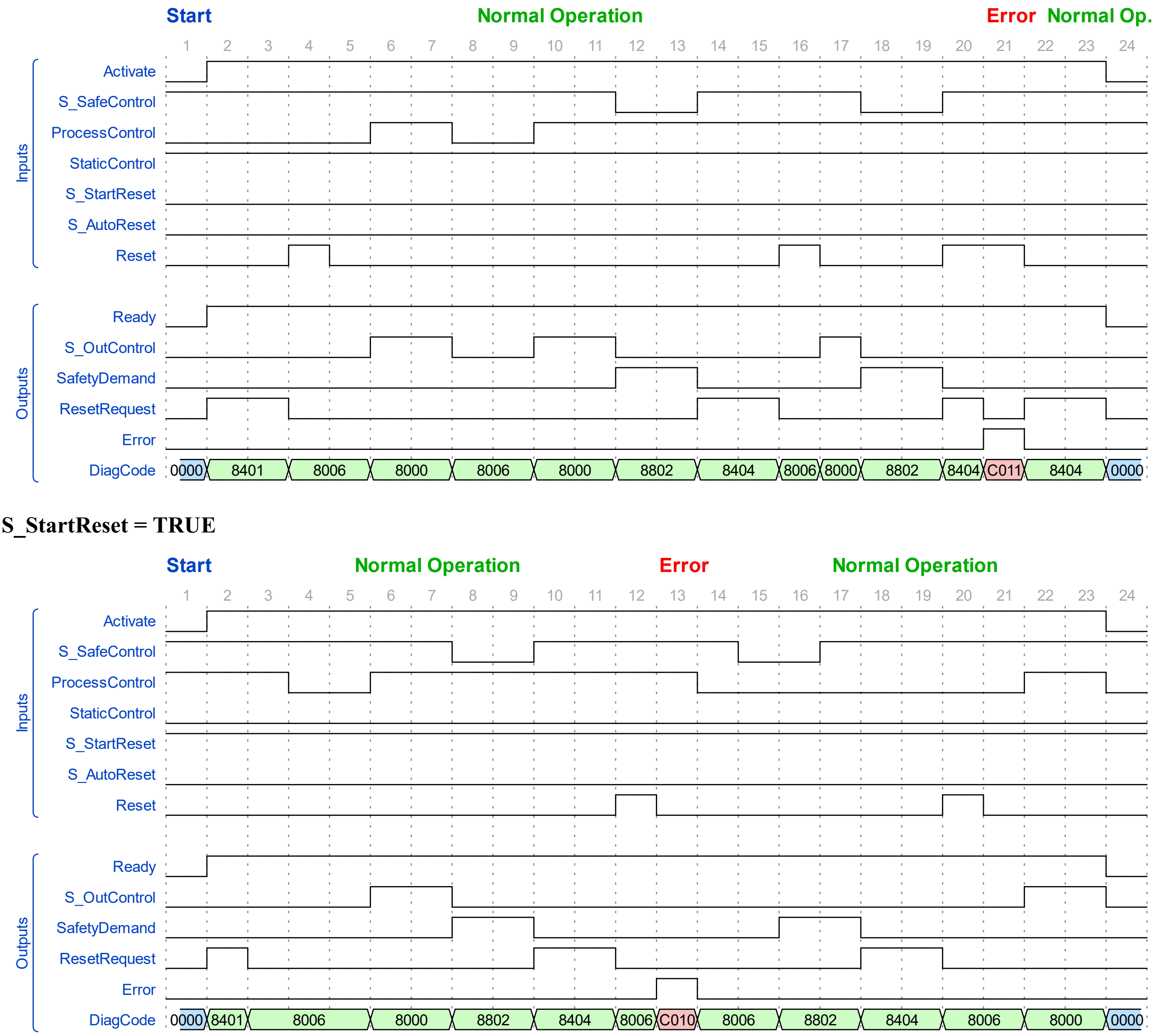


Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Figure 70: State diagram for SF_OutControl

Typical Timing Diagrams

S_StartReset = FALSE



S_StartReset = TRUE

Figure 71: Timing diagram for SF_OutControl

7.2.4 Error Detection

The following conditions force a transition to the Error state:

- Invalid static Reset signal in the process.
- Invalid static ProcessControl signal.
- ProcessControl and Reset are incorrectly interconnected due to programming error.

7.2.5 Error Behavior

In the event of an error, the S_OutControl output is set to FALSE and remains in this safe state.
To leave the Reset, Init or Lock error states, the Reset input must be set to FALSE. To leave the Control error state, the ProcessControl input must be set to FALSE.
After transition of S_SafeControl to TRUE, the optional startup inhibit can be reset by a rising edge at the Reset input.
After block activation, the optional startup inhibit can be reset by a rising edge at the Reset input.

7.2.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C001	Reset Error 1	Static Reset signal in state 8401.
		Ready = TRUE
		S_OutControl = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C011	Reset Error 2	Static Reset signal in state 8404.
		Ready = TRUE
		S_OutControl = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C010	Control Error	Static signal at ProcessControl in state 8006.
		Ready = TRUE
		S_OutControl = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C020	Init Error	Simultaneous rising trigger at Reset and ProcessControl in state 8401.
		Ready = TRUE
		S_OutControl = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C030	Lock Error	Simultaneous rising trigger at Reset and ProcessControl in state 8404.
		Ready = TRUE
		S_OutControl = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_OutControl = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8401	Init	Block activation startup inhibit is active. Reset required.
		Ready = TRUE
		S_OutControl = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

8802	Safe	Triggered safety function.
		Ready = TRUE
		S_OutControl = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

DiagCode	State Name	State Description and Output Setting

8404	Lock	Safety function startup inhibit is active. Reset required.
		Ready = TRUE
		S_OutControl = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT Reset
		Error = FALSE

8006	Output Disable	Process control is not active.
		Ready = TRUE
		S_OutControl = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8000	Output Enable	Process control is active and safety is enabled.
		Ready = TRUE
		S_OutControl = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

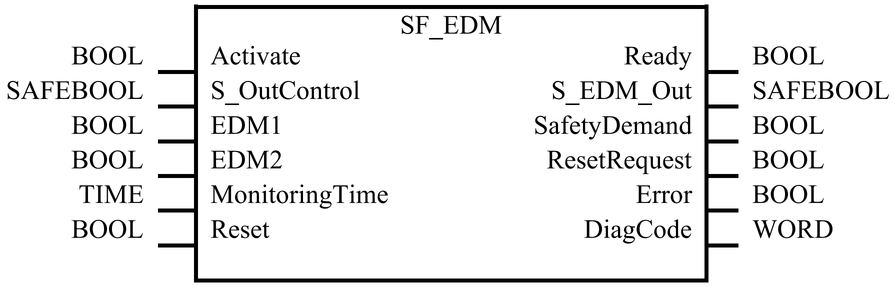
7.3 External Device Monitoring

7.3.1 Applicable Safety Standards

Standards	**Requirements**
IEC 60204-1:2016	Section 9.2.2: Stop function categories; Category 0
EN ISO 13849-1:2015	5.2.1 Safety-related stop function A safety-related stop function (e.g., initiated by a safeguard) shall, as soon as necessary after actuation, put the machine in a safe state. 6.2 Specifications of categories Fault detection (of the actuator, e.g., open circuits)
ISO 12100: 2010	6.2.11.2 Starting of an internal power source/switching on an external power supply. 6.2.11.4 Restart after power interruption
EN ISO 13849-1:2015	5.2.2 Manual reset function

7.3.2 Interface Description

FB Name	**SF_EDM**		
External device monitoring – The FB controls a safety output and monitors controlled actuators, e.g., subsequent contactors			
VAR_INPUT			
Name	*Data Type*	*Initial Value*	*Description, Parameter Values*
Activate	BOOL	FALSE	See Section 5.1.1 General Input Parameters
S_OutControl	SAFEBOOL	FALSE	Variable. Control signal of the preceeding safety FB's. Typical function block signals from the library (e.g., SF_OutControl, SF_TwoHandControlTypeII, and/or others). FALSE: Disable safety output (S_EDM_Out). TRUE: Enable safety output (S_EDM_Out).
EDM1	BOOL	FALSE	Variable. Feedback signal of the first connected actuator. FALSE: Switching state of the first connected actuator. TRUE: Initial state of the first connected actuator.
EDM2	BOOL	FALSE	Variable. Feedback signal of the second connected actuator. According to the actuators installed, the wiring between the feedback signals and the targeted safety level, it can be that only combined input is used here. In that case the user must use a graphic connection to jumper the EDM1 and EDM2 parameters. EDM1 and EDM2 are then controlled by the same signal. FALSE: Switching state of the second connected actuator. TRUE: Initial state of the second connected actuator.
MonitoringTime	TIME	#0ms	Constant. Max. response time of the connected and monitored actuators.
Reset	BOOL	FALSE	See Section 5.1.1 General Input Parameters
VAR_OUTPUT			
Ready	BOOL	FALSE	See Section 5.1.2 General Output Parameters
S_EDM_Out	SAFEBOOL	FALSE	Controls the actuator. The result is monitored by the feedback signal EDMx. FALSE: Disable connected actuators. TRUE: Enable connected actuators.
SafetyDemand	BOOL	FALSE	See 5.1.2 General Output Parameters
ResetRequest	BOOL	FALSE	See 5.1.2 General Output Parameters
Error	BOOL	FALSE	See Section 5.1.2 General Output Parameters
DiagCode	WORD	16#0000	See Section 5.1.2 General Output Parameters
Notes: -			



7.3.3 Functional Description

General:

The SF_EDM FB controls a safety output and monitors controlled actuators.

This function block monitors the initial state of the actuators via the feedback signals (EDM1 and EDM2) before the actuators are enabled by the FB.

The function block monitors the switching state of the actuators (MonitoringTime) after the actuators have been enabled by the FB.

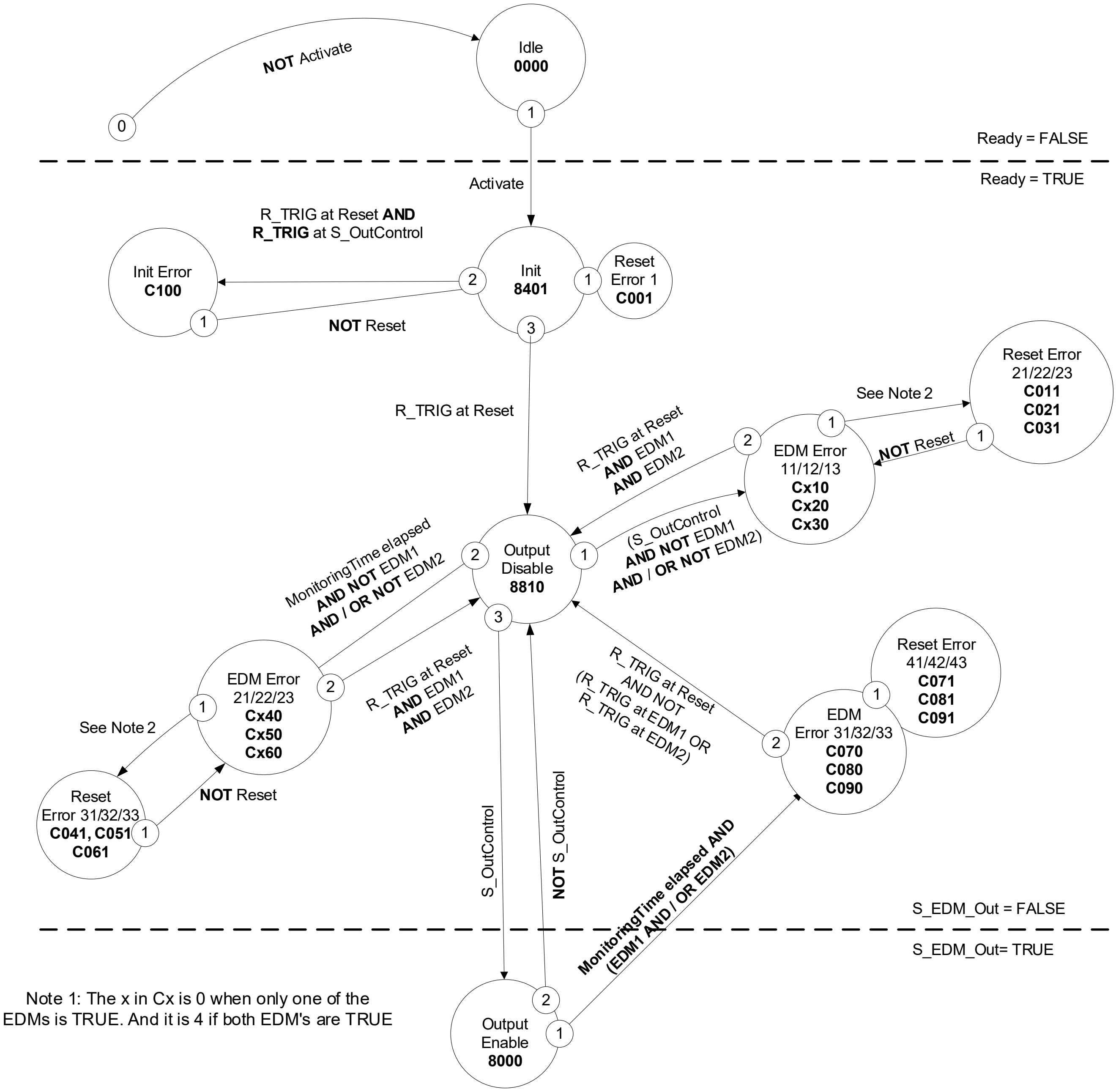
Two single feedback signals must be used for an exact diagnosis of the connected actuators. A common feedback signal from the two connected actuators must be used for a restricted yet simple diagnostic function of the connected actuators. When doing so, the user must connect this common signal to both parameter EDM1 and parameter EDM2. EDM1 and EDM2 are then controlled by the same signal.

The switching devices used in the safety function should be selected from the category specified in the risk analysis (EN ISO 13849-1).

Optional startup inhibits:

- Startup inhibit in the event of block activation.

State Diagram



Note 1: The x in Cx is 0 when only one of the EDMs is TRUE. And it is 4 if both EDM's are TRUE

Note 2:
Condition for transition Cx10 (EDM Error 11) to C011 (Reset Error 21) and Cx40 (EDM Error 21) to C041 (Reset Error 31) :
(Reset AND NOT R_TRIG at Reset AND EDM1 AND EDM2) OR (R_TRIG at Reset AND R_TRIG at EDM1)

Condition for transition Cx20 (EDM Error 12) to C021 (Reset Error 22) and Cx50 (EDM Error 22) to C051 (Reset Error 32) :
(Reset AND NOT R_TRIG at Reset AND EDM1 AND EDM2) OR (R_TRIG at Reset AND R_TRIG at EDM2)

Condition for transition Cx30 (EDM Error 13) to C031 (Reset Error 23) and Cx60 (EDM Error 23) to C061 (Reset Error 33) :
(Reset AND NOT R_TRIG at Reset AND EDM1 AND EDM2) OR (R_TRIG at Reset AND R_TRIG at EDM1 AND R_TRIG at EDM2)

Note: The transition from any state to the Idle state due to Activate = FALSE is not shown. However, these transitions have the highest priority (0).

Figure 72: State diagram for SF_EDM

Typical Timing Diagrams

S_StartReset = FALSE

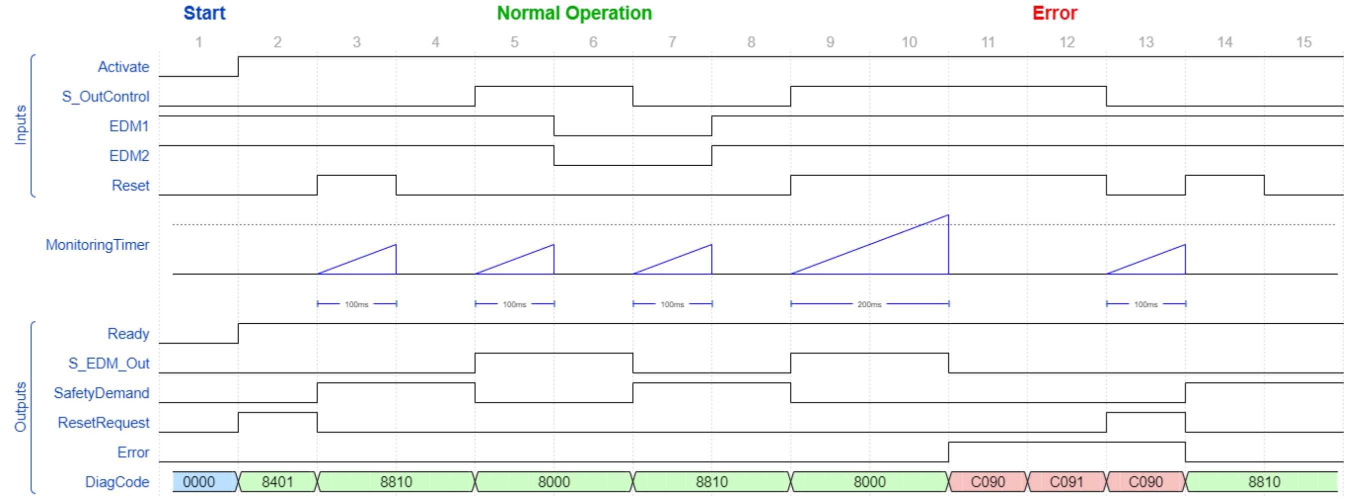


Figure 73: Timing diagram for SF_EDM

7.3.4 Error Detection

The following conditions force a transition to the Error state:

- Invalid static Reset signal in the process.
- Invalid EDM signal in the process.
- S_OutControl and Reset are incorrectly interconnected due to programming error.

7.3.5 Error Behavior

In error states, the outputs are as follows:

- In the event of an error, the S_EDM_Out is set to FALSE and remains in this safe state.
- An EDM error message must always be reset by a rising trigger at Reset.
- A Reset error message can be reset by setting Reset to FALSE.

After block activation, the optional startup inhibit can be reset by a rising edge at the Reset input.

7.3.6 Function Block-Specific Error and Status Codes

FB-specific error codes:

DiagCode	State Name	State Description and Output Setting

C001	Reset Error 1	Static Reset signal in state 8401.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C011	Reset Error 21	Static Reset signal or same signals at EDM1 and Reset (rising trigger at Reset and EDM1 at the same time) in state C010.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C021	Reset Error 22	Static Reset signal or same signals at EDM2 and Reset (rising trigger at Reset and EDM2 at the same time) in state C020.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

C031	Reset Error 23	Static Reset signal or same signals at EDM1, EDM2, and Reset (rising trigger at Reset, EDM1, and EDM2 at the same time) in state C030.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C041	Reset Error 31	Static Reset signal or same signals at EDM1 and Reset (rising trigger at Reset and EDM1 at the same time) in state C040.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C051	Reset Error 32	Static Reset signal or same signals at EDM2 and Reset (rising trigger at Reset and EDM2 at the same time) in state C050.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C061	Reset Error 33	Static Reset signal or same signals at EDM1, EDM2, and Reset (rising trigger at Reset, EDM1, and EDM2 at the same time) in state C060.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C071	Reset Error 41	Static Reset signal in state C070.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C081	Reset Error 42	Static Reset signal in state C080.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C091	Reset Error 43	Static Reset signal in state C090.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C010	EDM Error 11	The signal at EDM1 is not valid in the initial actuator state. In state 8810 the EDM1 signal is FALSE when enabling O_OutControl.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = R[1]
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

C020	EDM Error 12	The signal at EDM2 is not valid in the initial actuator state. In state 8810 the EDM2 signal is FALSE when enabling O_OutControl.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = R[1]
		Error = TRUE

C030	EDM Error 13	The signals at EDM1 and EDM2 are not valid in the initial actuator states. In state 8810 the EDM1 and EDM2 signals are FALSE when enabling O_OutControl.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

C040	EDM Error 21	The signal at EDM1 is not valid in the initial actuator state. In state 8810 the EDM1 signal is FALSE and the monitoring time has elapsed.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = R[1]
		Error = TRUE

C050	EDM Error 22	The signal at EDM2 is not valid in the initial actuator state. In state 8810 the EDM2 signal is FALSE and the monitoring time has elapsed.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = R[1]
		Error = TRUE

C060	EDM Error 23	The signals at EDM1 and EDM2 are not valid in the initial actuator states. In state 8810 the EDM1 and EDM2 signals are FALSE and the monitoring time has elapsed.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = R[1]
		Error = TRUE

C070	EDM Error 31	The signal at EDM1 is not valid in the actuator switching state.
		In state 8000 the EDM1 signal is TRUE and EDM2 is FALSE and the monitoring time has elapsed.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT RESET
		Error = TRUE

C080	EDM Error 32	The signal at EDM2 is not valid in the actuator switching state.
		In state 8000 the EDM2 signal is TRUE and EDM1 is FALSE and the monitoring time has elapsed.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT RESET
		Error = TRUE

DiagCode	State Name	State Description and Output Setting

C090	EDM Error 33	The signals at EDM1 and EDM2 are not valid in the actuator switching state. In state 8000 both the EDM1 and EDM2 signals are TRUE and the monitoring time has elapsed.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = NOT RESET
		Error = TRUE

C100	Init Error	Similar signals at S_OutControl and Reset (R_TRIG at same cycle) detected (may be a programming error)
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = TRUE

FB-specific status codes (no error):

DiagCode	State Name	State Description and Output Setting

0000	Idle	The function block is not active (initial state).
		Ready = FALSE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

8401	Init	Block activation startup inhibit is active. Reset required.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = FALSE
		ResetRequest = TRUE
		Error = FALSE

8810	Output Disable	EDM control is not active. Timer starts when state is entered.
		Ready = TRUE
		S_EDM_Out = FALSE
		SafetyDemand = TRUE
		ResetRequest = FALSE
		Error = FALSE

8000	Output Enable	EDM control is active. Timer starts when state is entered.
		Ready = TRUE
		S_EDM_Out = TRUE
		SafetyDemand = FALSE
		ResetRequest = FALSE
		Error = FALSE

Note:
R[1]: IF EDM_1 = TRUE AND
EDM_2 = TRUE THEN
R = NOT RESET

ELSE
R = FALSE

Appendix 1. Compliance Procedure and Compliance List

Listed in this Appendix are the requirements for the compliance statement from the supplier of the safety specification. The compliance statement consists of two main groups:

- Reduction in programming languages and functionality (see "Appendix 1.2 Applicable reductions in the Development Environment “)
- The definition of a set of function blocks with safety-related functionality (see "Appendix 1.3 Overview of the supported Function Blocks").

The supplier must fill out the tables for their implementation, according to their product, committing their support to the specification itself.

By submitting these tables to PLCopen, and following approval by PLCopen, the list will be published on the PLCopen website (http://www.PLCopen.org) as specified in "Appendix 2 The PLCopen Safety Logo and Its Use" below.

In addition to this approval, the supplier is provided with access and usage rights for the PLCopen Safety logo, as described in Appendix 2 The PLCopen Safety Logo and Its Use.

Appendix 1.1. Supplier Statement

Supplier name	
Supplier address	
City	
Country	
Phone	
Fax	
Website	
Product name	
Product version	
Release date	
Certified by	

I hereby state that the following tables as filled out and submitted correspond to our product and the accompanying user manual, as stated above.

Name of representative:

Date of signature (dd/mm/yyyy):

Signature:

Appendix 1.2. Applicable reductions in the Development Environment

Supported User Levels (See Section 4)	**Supported**	**Comments (< 48 Characters)**
Basic level		
Extended level		
System level		How is it supported?

Table 7: Supported user levels

Supported Programming Languages	**Supported**	**Comments (< 48 Characters)**
Function Block Diagram, FBD		
Ladder Diagram, LD		
Structured Text, ST		

Table 8: Supported programming languages

Supported Data Types	**Supported**	**Comments (< 48 Characters)**
SAFEBOOL		
ANY_SAFEREAL		Which?
ANY_SAFEINT		Which?
ANY_SAFEDURATION		Which?
ANY_SAFEBIT		
ANY_SAFEDATE		
BOOL		
INT		
DINT		
REAL		
WORD		
TIME		
DURATION		
DATE		
Structures data type		See 4.3 Reduction in Data Types and Declarations

Table 9: Supported data types

Supported Functions and FBs	**Supported Basic Level**	**Supported Ext.Level**	**Comments (< 48 Words)**
AND			
OR			
XOR, NOT			
ADD, MUL, SUB, DIV, MOD, EXPT +, *, -, /, MOD, **			
NEG, -			
EQ, NE, =, <>			
GT, GE,LE, LT >, >=, <=, <			
SEL, MAX, MIN, LIMIT, MUX			
Type Conversion functions			Specifiy which
Time functions			Specifiy which
Unary REAL functions			Specifiy which
TON			
TOF			
TP			
CTU			
CTD			
CTUD			
Bistable FB (SR, RS)			

Edge detection			
Others?			Specifiy which

Table 10: Supported Functions and Function Blocks at Basic Level

Description	Supported at Extended Level	Comments
(expression)		
Identifier (argument list)		
A := B; CV := CV+1; C:= ABS(X);		
Function Block Instance (…)		
RETURN;		

IF …
THEN …
ELSIF …
THEN …
ELSE …
END_IF		

CASE … OF
…
ELSE …
END_CASE		

FOR … TO … BY … DO … END_FOR		
EXIT		
CONTINUE		
Others?		

Table 11: Supported functionality of ST at Extended Level

Appendix 1.3. Overview of the supported Function Blocks

Function Blocks	Supported	Comments (<= 48 Characters)
SF_ResetButton		
SF_Equivalent		
SF_Antivalent		
SF_ModeSelector		
SF_EmergencyStop		
SF_ESPE		
SF_PSE		
SF_TwoHandControlTypeII		
SF_TwoHandControlTypeIII		
SF_TestableSafetySensor		
SF_MutingSeq		
SF_MutingPar		
SF_MutingPar_2Sensors		
SF_EnableSwitch		
SF_EnableSwitch_2		
SF_Guard		
SF_GuardLocking_2		
SF_GuardLockingSerial		
SF_Override		
SF_SafetyRequest		
SF_OutControl		
SF_EDM		

Table 12: Overview of the function blocks

Appendix 2. The PLCopen Safety Logo and Its Use

For quick identification of compliant products, PLCopen has developed a logo for the Safety Specification:

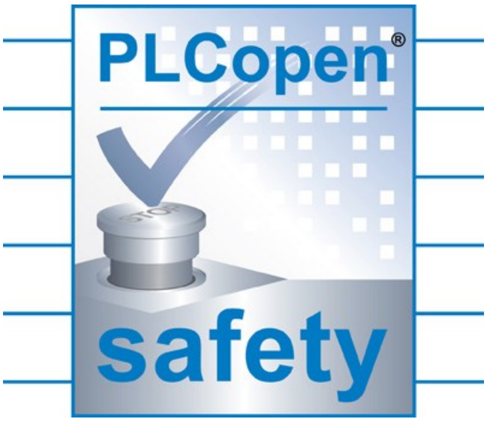


Figure 74: The PLCopen Safety logo

This logo is owned and trademarked by PLCopen.

In order to use this logo free of charge, the relevant company must meet all of the following requirements:

1. The company must be a voting member of PLCopen.
2. The company must comply with the existing specification, as specified by the PLCopen Technical Committee 5 - Safety, and as published by PLCopen, and of which this statement is a part.
3. This compliance is submitted in writing by the company to PLCopen, clearly stating the applicable software package and the supporting elements of all the specified tables, as specified in this document.
4. The company is aware that this compliance is only a statement of the supporting elements as specified in this document. In particular, the company is aware that this statement does not have any relationship to the implementation itself, nor the fulfillment of any requirements as specified in any safety standard, safety procedure, or development procedure, and does not state anything regarding the quality of the product itself, nor certification procedures performed by a third party.
5. In the event of non-fulfillment, which must be decided by PLCopen, the company will receive a written statement to this effect from PLCopen. The company will have a period of one month to either adapt their software package in such a way that it is compliant, i.e., by issuing a new compliance statement, or removal of all reference to the specification, including the use of the logo, from all their specifications, be they technical or promotional material.
6. The logo must be used as is - i.e., in its entirety. It may only be altered in size if the original scale and color settings are maintained.
7. The logo must be used in the context of PLCopen Safety.