Safety Drive Profile

Generic Safety Drive Profile for adjustable speed electrical power drive systems that are suitable for use in safety-related application PDS(SR)

Document: ETG.6100.2 S (R) V1.2.0

Nomenclature:

ETG-Number	ETG.6100.2
Type	S (Standard)
State	R (Release)
Version	1.2.0

Created by: ETG
Contact: info@ethercat.org
Date: 08.02.2016

LEGAL NOTICE

DOCUMENT HISTORY

Version	Comment
1.0.0	First release
1.0.2	Name of parameters changed – more meaningful Prioritization of Stop functions, parallel activation of monitoring functions
1.0.3	Figure 22 (SDI) Position graph corrected Changed SIGNEDxx to INTEGERxx Hex values representation: change from 0x1000 to 1000_h Access rights of objects is defined in Safety communication specific part → deleted
1.1.0 D	Draft Release
1.2.0 R	Release version SMS: change data type for SMS objects to INTEGERxx

CONTENTS

TABLES

FIGURES

ABBREVIATIONS

C	Conditional
ETG	EtherCAT Technology Group
IEC	International Electrotechnical Commission
M	Mandatory
MSB	Most Significant Bit
O	Optional

1 Scope

This document, ETG.6100.2, defines a generic safety drive profile for a drive with integrated safety functionality. It defines the process data structure and the safety objects in the drive for the configuration of various safety functions according to the IEC 61800-5-2.

2 References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Other References

[1] IEC 61800-7: Adjustable speed electrical power drives systems – Part 7: Generic interface and use of profiles for power drive systems

[2] IEC 61800-5-2, Adjustable speed electrical power drive systems, Part 5: Safety Requirements – Section 2: Functional

3 Terms, Definitions and Word Usage

3.1 Word usage: shall, should, may, can

The word *shall* is used to indicate mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*).

The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted to*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

The elements of an object are described in a form shown in Table 1.

Table 1: Example for object description

Object Code	Name		Type	M/O/C	Value
ARRAY	t_SS1		UNSIGNED16	C	Time delay to initiate STO [Time Units]
	0	Number of entries	UNSIGNED8	M	1…254
	1	t_SS1_1	UNSIGNED16	M	1st Instance of function
	2	t_SS1_2	UNSIGNED16	O	2nd Instance of function
	…	…			…
	254	t_SS1_254	UNSIGNED16	O	254th Instance of function

Object code
Object code describes the type of the object. Allowed types
VAR: Single Variable
ARRAY: Array of variables of the same Type
RECORD: Struct of variables of different Types

Name
Name is a unique name string for this object. If this object is of object code ARRAY or RECORD it consists of several elements (called subindex). These elements are listed in the following rows. The number denotes the subindex number. The first element with number 0 always contains the number of following entries.

Type
Type denotes the data type of this element.

M/O/C
M/O/C indicates whether the element is mandatory (M), optional (O) or conditional (C) that means it depends upon setting of other attributes. Within the subindex elements the (M) means this subindex is mandatory if the object exists.

NOTE: Many of these objects are conditional. That means if the corresponding function is supported this object is mandatory.

Value
Value denotes the allowed values and/or a description of the meaning.

3.1.1 Syntax conventions for instances of functions and objects

Most functions defined in this specification can be implemented in **several instances** within a device.

EXAMPLE According to the current area of operation the drive can have different speed limits. Therefore the function SLS can be implemented e.g. in four instances (SLS1, SLS2, SLS3, SLS4). These four instances are configured to four different operation areas. The switching between these areas can easily be done via four bits in the safety controlword.

To support a unique definition of the parameter of these instances, the objects are defined as ARRAY. Subindex 1 of these objects defines the parameter for the first instance, subindex 2 the parameter of the second instance and so forth. Up to 254 instances of each function can be defined.

For a better overview, the subindices for the several instances are not shown in the object specifications in chapter 7, but only the objects specification itself. The example shown in Table 1 is therefore defined as shown in Table 2.

Table 2: Short object description for specification

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	t_SS1	UNSIGNED16	C	Time delay to initiate STO [Time Units]

3.1.2 Data Type of position, velocity and acceleration values

All position, velocity and acceleration values are defined as 32-Bit objects and as 16-Bit objects. The corresponding parameters are defined as 32- and 16-Bit values, too.

4 Functional Principle

4.1 General

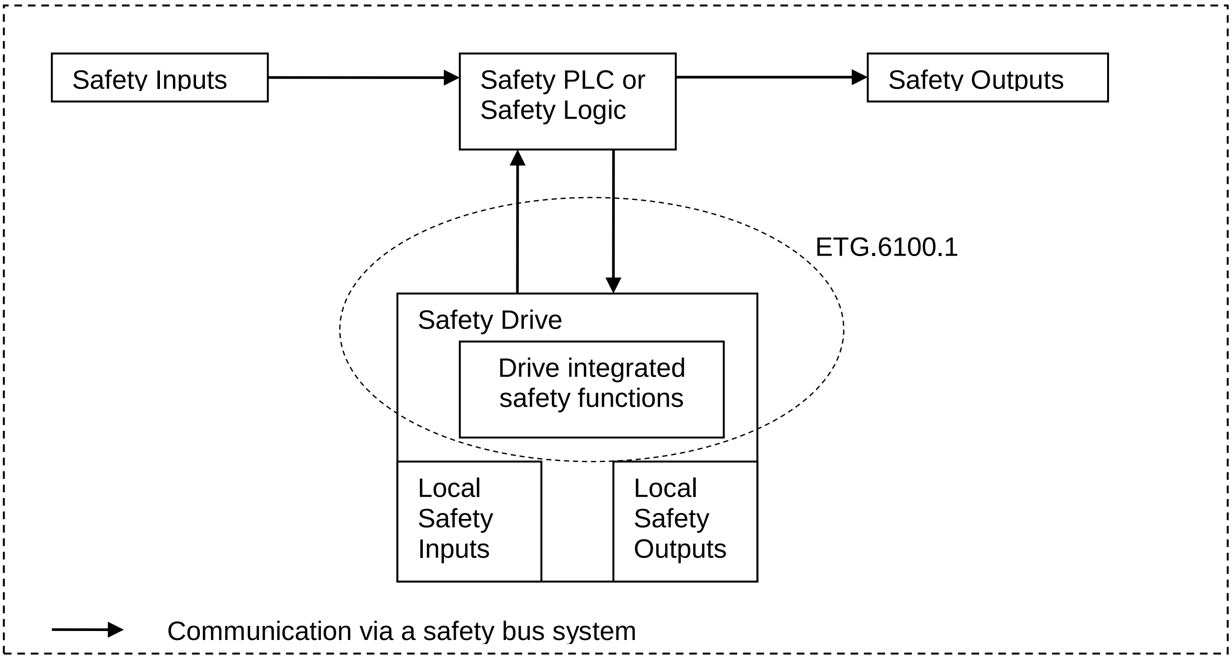

Figure 1: Safety Drive within a Safety System

In a safety system with safety logic the safety drive gets safety commands from the safety logic. These commands should activate integrated safety functions in the drive. The drive should send status information to the safety logic, so that the safety logic can react according to the status of the drive, e.g. to enable a following action (e.g. open a safety gate).

The safety drive can also provide local safety inputs and safety outputs. The safety inputs can be used to activate internal safety functions or can be assigned to the safety status information. The safety outputs can be used to assign safety status information or can be set by the command information of the safety logic. This is outside the scope of this specification.

Note: The safety functions or parts of the safety functions can also be realized in the safety logic.

4.2 Safety functions

The safety drive shall implement one or more of the following safety functions:

Stopping functions

- Safe Torque Off (STO)
- Safe Stop 1 (SS1)
- Safe Stop 2 (SS2)

Other safety functions

- Safe Operating Stop (SOS)
- Safe Brake Control (SBC)
- Safe Speed Range (SSR)
- Safe Acceleration Range (SAR)
- Safely-limited Speed (SLS)
- Safely-limited Position (SLP)
- Safe Maximum Speed (SMS)
- Safely-limited Torque (SLT)

- Safe Maximum Acceleration (SMA)
- Safely-limited Increment (SLI)
- Safe Direction (SDI)
- Safe Speed Monitor (SSM)
- Safe CAM (SCA)

4.3 Parallel activation of safety functions

Parallel activation of safety functions shall be done without influencing each other.

Note 1: Since those functions are monitoring functions they can be active in parallel. Even instances of the same function can be active in parallel – this will result that the more restrictive limit will be observed.

Note 2: Activation can be done via the safety fieldbus or via local safety inputs.

5 Process Data

5.1 General

For a safety drive a *safety controlword* and a *safety statusword* is defined in this specification. Manufactor specific additional safety controlwords and safety statuswords can be defined.

5.2 Safety controlword

5.2.1 General

The *safety controlword* shall activate the corresponding drive integrated safety functions. The *safety controlword* consists of one byte with a default mapping and optional additional bytes with manufacturer specific mapping.

5.2.2 Safety controlword, 1st byte (default mapping)

The first byte of the *safety controlword* is defined in Table 3.

Table 3: Safety controlword: 1st byte

Bit	Name	Description
0	STO	Safe Torque Off 0: activate 1: deactivate
1	SS1_1	Safe Stop 1, Instance 1 0: activate 1: deactivate
2	SS2_1	Safe Stop 2, Instance 1 0: activate 1: deactivate
3	SOS_1	Safe Operating Stop, Instance 1 0: activate 1: deactivate
4	SSR_1	Safe Speed Range, Instance 1 0: activate 1: deactivate
5	SDIp	Safe Direction positive 0: disable positive direction 1: enable positive direction
6	SDIn	Safe Direction negative 0: disable negative direction 1: enable negative direction
7	Error Ack	Error Acknowledge 0: no acknowledge 0→1: Acknowledge of an error in the drive

The implementation of all bits is optional, i.e. not all safety functions have to be implemented. If a function is supported the corresponding bit and the behavior of the function shall be used as described in this specification.

The bits in the *safety controlword*, which are not supported by the safety drive, should be set to "0" by the safety logic.

5.2.3 Safety controlword, further bytes (manufacturer specific)

In addition to the 1st byte of the *safety controlword*, the manufacturer can implement up to 31 additional Control Bytes. These bytes contain (other) instances of the described safety functions or instances of manufacturer specific safety functions.

The bits in the manufacturer specific safety controlwords, which are not supported in the safety drive, should be set to "0" by the safety logic.

The definition of the bits shall follow the same principal for activation: “0” = activate, “1” = deactivate.

If these bits are assigned to local safety outputs the following meaning is specified:
"0" = output is off (safe), "1" = output is on.

5.2.4 Objects

The following object is defined:

Table 4: Object safety controlword

Object Code	Name		Type	M/O/C	Value
ARRAY	safety controlword		UNSIGNED8	O	
	0	Number of entries	UNSIGNED8	M	1…32
	1	controlword 1st byte	UNSIGNED8	M	1st byte of the safety controlword (default mapping, see Table 3)
	2	controlword 2nd byte	UNSIGNED8	O	2nd byte of the safety controlword (manufacturer specific)
	…	…			…
	32	controlword 32nd byte	UNSIGNED8	O	32nd byte of the safety controlword (manufacturer specific)

5.3 Safety statusword

5.3.1 General

The *safety statusword* reflects the current state of the safety functions.

The *safety statusword* consists of one byte with a default mapping and optional additional bytes with manufacturer specific mapping.

5.3.2 Safety statusword, 1st byte (default mapping)

The first byte of the *safety statusword* is defined in Table 5

Table 5: Safety statusword, 1st byte (fix meaning)

Bit	Name	Description
0	STO active	Safe Torque off 0: is not active 1: is active
1	SSM_1	Safe Speed Monitor 1, Instance 1 0: the drive is moving faster than the limit n_UL_SSM_1 1: the drive is moving slower or equal than the limit n_LL_SSM_1
2	SSM_2	Safe Speed Monitor 2, Instance 2 0: the drive is moving faster than the limit n_UL_SSM_2 1: the drive is moving slower or equal than the limit n_LL_SSM_2
3	SOS_1 active	Safe Operating Stop, Instance 1 0: is not active 1: is active
4	SSR_1 active	Safe Speed Range, Instance 1 0: is not active 1: is active and speed of the drive is within the safe speed range
5	SDIp active	Safe Direction positive 0: Drive does not move in positive direction 1: Drive moves in positive direction
6	SDIn active	Safe Direction negative 0: Drive does not move in negative direction 1: Drive moves in negative direction
7	Error	Error 0: no Error 1: at least one safety error has occurred

The implementation of all bits is optional. Not all status information has to be provided. If a function is supported the corresponding status bit and the behavior of the function shall be used as described in this specification.

The bits in the safety statusword, which are not supported by the safety drive, shall be set to "0" by the drive.

NOTE: The Bits SS1 and SS2 are not part of the safety statusword. These functions consist of two parts: First part is the deceleration; second part is the stopping function. During deceleration the status SS1 and SS2 can be part of standard status information. The drive is still in moving, therefore this information is not safety related. The resulting stop functions (STO, SOS) are part of the *safety statusword*.

5.3.3 Safety statusword, further bytes (manufacturer specific)

In addition to the 1st byte of the safety statusword, the manufacturer can implement up to 31 additional safety status bytes. These bytes contain (other) instances of the described safety functions or instances of manufacturer specific safety functions.

The bits in the manufacturer specific safety statusword, which are not supported in the safety drive, shall be set to "0" by the drive.

The definition of the bits shall follow the same principle for activation: “0” = is not active, “1” is active.

If these bits are assigned to local safety inputs the following meaning is specified:
"0" = input is off (safe), "1" = input is on.

Note It is also possible to give a feedback for each selected safety function in the standard communication of the drive, whether the function is selected. This feedback is not safety relevant, but it may be used in the general application, e.g. to initiate drive functions or for diagnosis.

5.3.4 Objects

The following object is defined:

Table 6: Object safety statusword

Object Code	Name		Type	M/O/C	Value
ARRAY	safety statusword		UNSIGNED8	O	
	0	Number of entries	UNSIGNED8	M	1…32
	1	statusword 1st byte	UNSIGNED8	M	1st byte of the safety statusword (Default mapping, see Table 5)
	2	statusword 2nd byte	UNSIGNED8	O	2nd byte of the safety statusword (manufacturer specific)
	…	…			…
	32	statusword 32nd byte	UNSIGNED8	O	32nd byte of the safety statusword (manufacturer specific)

5.4 Safety process values

5.4.1 General

The PDS(SR) can send safety-related information that are calculated within the drive integrated safety function. These values can be used for example by the safety logic to calculate safety limiting functions.

5.4.2 Objects

The following objects are defined

Table 7: Safety-related input objects

Object Code	Name	Type	M/O/C	Value
VAR	safe position actual value i16	INTEGER16	O	Safety-related calculated position actual value
VAR	safe position actual value i32	INTEGER32	O	Safety-related calculated position actual value

Object Code	Name	Type	M/O/C	Value
VAR	safe velocity actual value i16	INTEGER16	O	Safety-related calculated velocity actual value
VAR	safe velocity actual value i32	INTEGER32	O	Safety-related calculated velocity actual value
VAR	safe acceleration actual value i16	INTEGER16	O	Safety-related calculated acceleration actual value
VAR	safe acceleration actual value i32	INTEGER32	O	Safety-related calculated acceleration actual value
VAR	safe torque actual value	INTEGER16	O	Safety-related calculated torque actual value

6 Unit definition

The values for time, position, velocity, acceleration and torque parameters are given in device-defined units. The parameter described in the safety functions refer to these unit-objects.

The definition of the unit objects for the following values is fieldbus specific:

- Time Unit
- Position Unit
- Velocity Unit
- Acceleration Unit
- Torque Unit

7 Restart and error behavior

7.1 Restart behavior

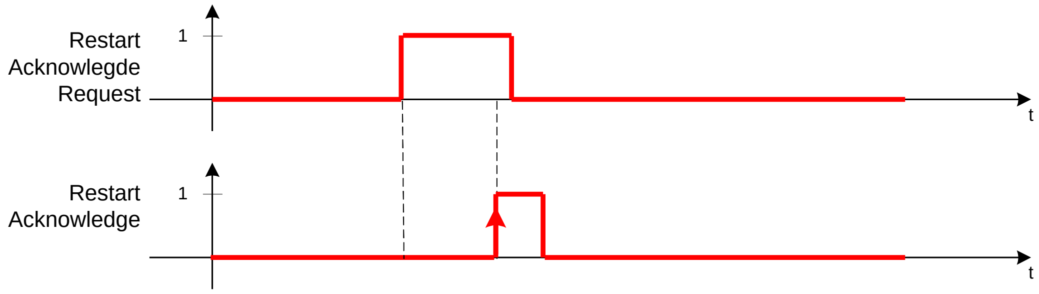

Figure 2: Restart acknowledge behavior

In some safety applications it is necessary that the drive prevents unintended movement of the machine after start-up or after a stop condition (STO, SS1, SS2).

The restart behaviour can be configured separately for STO (see 8.1) and SS2 (see 8.5). The STO behaviour is used for STO and SS1, the SS2 behaviour is used for SS2.

NOTE: SS2 will end in state SOS. The restart condition has to be taken into account for the activated SOS instance, if it is performed as a result of a SS2 stop function.

If the drive signals a “Restart acknowledge request” in the safety statusword, a rising edge in the “Restart acknowledge” of the safety controlword gives the acknowledge signal to (re-)start the drive.

When the drive clears the “Restart acknowledge request”, the “Restart acknowledge” should be released, too.

7.1.1 Objects for restart behavior

The following object is defined:

Table 8: Object restart acknowledge

Object Code	Name	Type	M/O/C	Value
VAR	restart acknowledge	BOOLEAN	O	Read: 0: "Restart acknowledge request" not pending 1: "Restart acknowledge request" pending Write 0→1: Restart acknowledge

7.2 Error behavior

7.2.1 Configurable error reactions

If the error reaction of a safety function is configurable the corresponding object (e.g. SSR error reaction) contains a link to the safety function that should be performed.

NOTE: Usually stopping functions (STO, SS1 and SS2) are used as error reactions.

The format of the error reaction object is fieldbus specific.

Table 9: Example for an error reaction object

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	XXX error reaction	UNSIGNED32	O	Reference to Error reaction command for violation of limits Default: *STO command* No Reaction: 0x0000 0000 Format is fieldbus specific

7.2.2 Error acknowledge

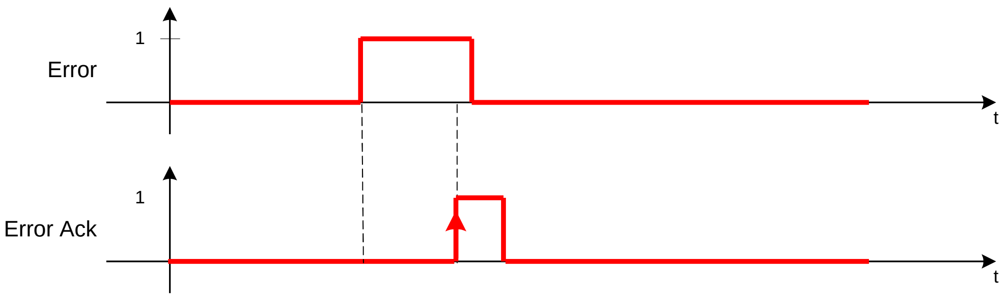

Figure 3: Error acknowledge behavior

Errors can occur in a safety function or in self-test functions or for manufacturer specific reasons. All error reasons that are defined in the following chapters shall set the error bit in the safety statusword.

The error reason should be given in the communication specific diagnosis handling.

If a drive signals an error by setting the "Error"-Bit in the safety statusword, a rising edge in the “Error acknowledge” Bit in the safety controlword gives the acknowledge to reset this error. With one acknowledge the error state is cleared (this means all errors at once). If the error state re-occurs immediately afterwards or can't be cleared, the "Error" bit in the statusword remains set.

7.2.2.1 Objects for error acknowledge

The following object is defined:

Table 10: Object for error acknowledge

Object Code	Name	Type	M/O/C	Value
VAR	error acknowledge	BOOLEAN	O	Read: 0: No error pending 1: At least one error pending Write 0→1: Error acknowledge

8 Safety Functions

8.1 Safe Torque Off STO

8.1.1 Description

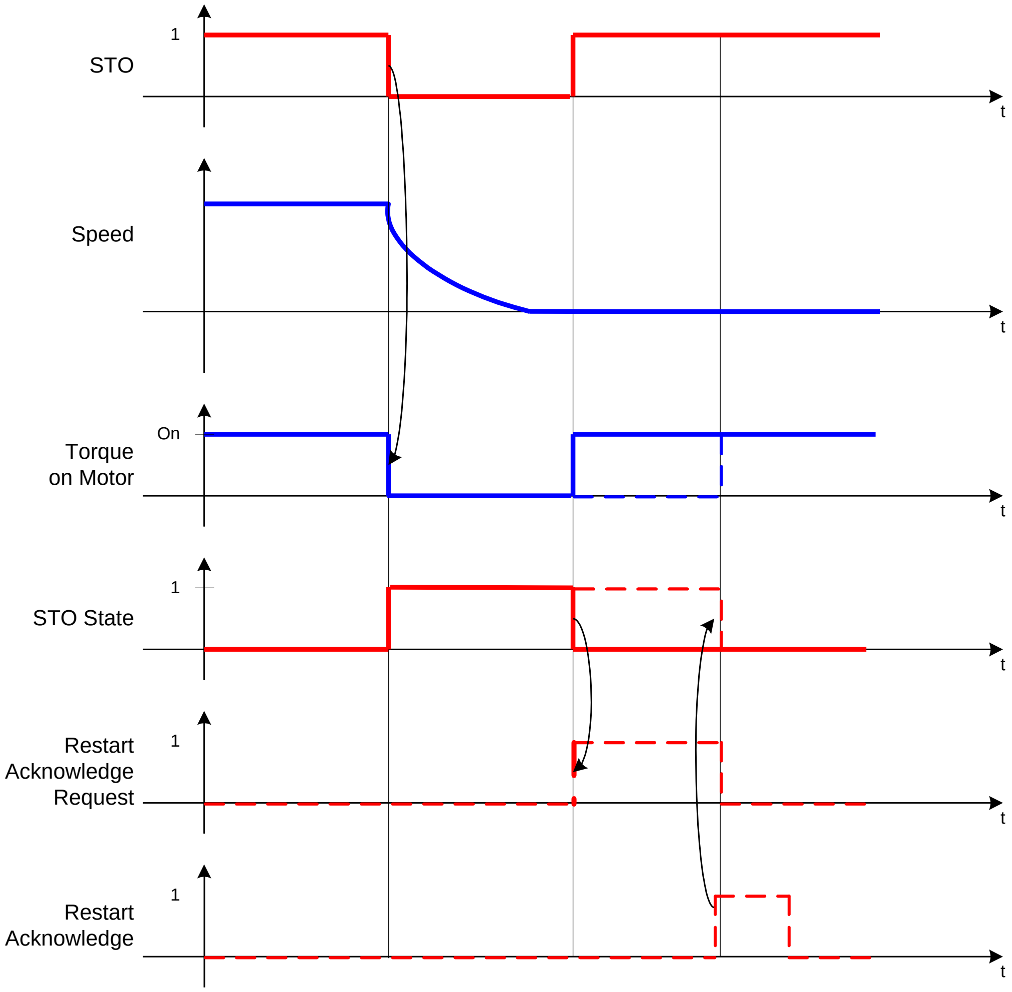

Figure 4: Functional principle of Safe Torque Off (STO)

Immediately after activation of STO the torque on the motor is switched off.

Note: In most cases there will be no galvanic isolation between the power and the motor. That means STO does not imply Emergency off functionality.

If the drive supports a local restart acknowledge than the STO state shall be set as long as the STO request Bit is set OR the restart request bit is set.

Optionally the function SBC can be activated to close the brakes (see 8.3).

8.1.2 Activation

Bit STO in the safety controlword

8.1.3 Objects for STO

Objects for STO are defined in Table 11.

Table 11: Objects for STO

Object Code	Name	Type	M/O/C	Value
VAR	STO command	BOOLEAN	O	Read: 0: is not active 1: is active Write: 0: activate 1: deactivate
VAR	STO restart acknowledge behavior	BOOLEAN	O	0: Restart acknowledge for STO switched-off (automatic restart) 1: Restart acknowledge for STO switched on
VAR	STO activate SBC	UNSINGED32	O	Reference to SBC instance (*SBC command*) that should be activated 0x0000 0000 if no brake is used Format is fieldbus specific

8.1.4 Error reaction

No error reaction.

8.2 Safe Stop 1 SS1

8.2.1 Description

8.2.1.1 SS1 with time monitoring

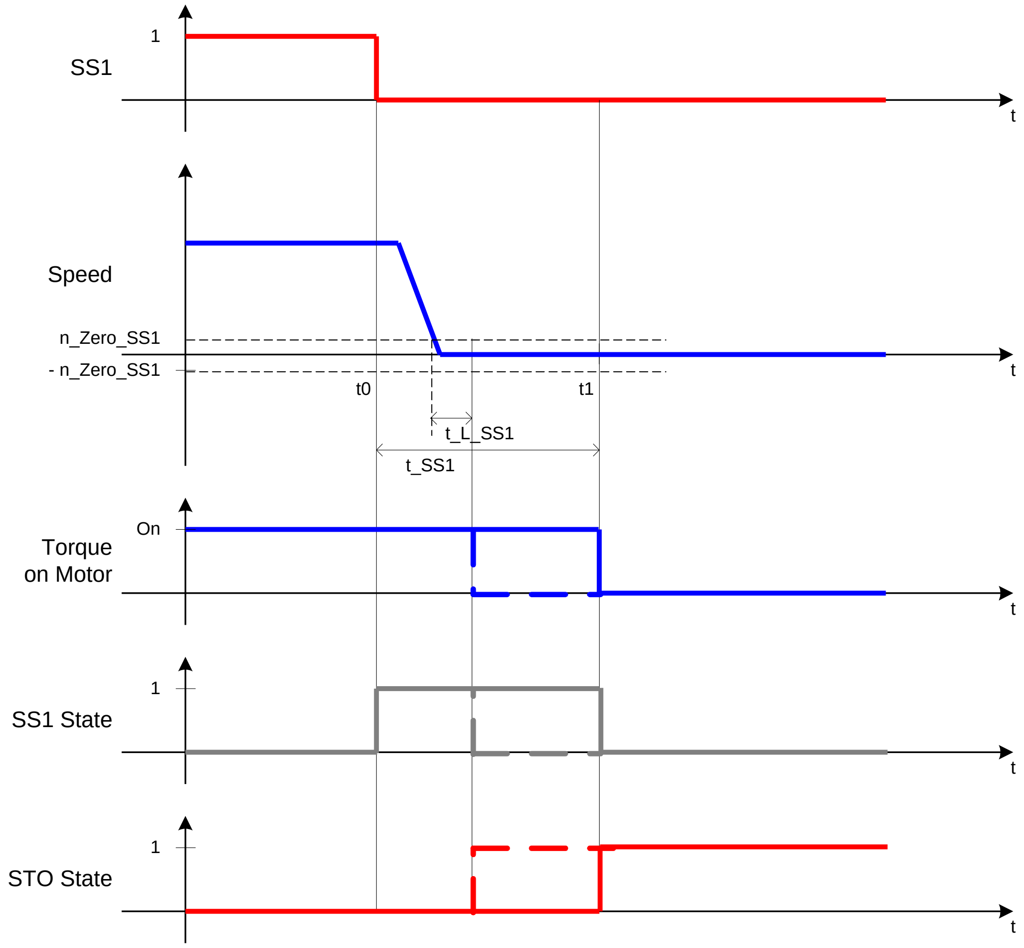

Figure 5: Functional principle of Safe Stop 1 (SS1) with time limit

With activation of SS1 the *SS1 time to STO* (t_SS1) is started. The PDS(SR) starts with deceleration (standard task) and at least after t_SS1, the STO state is initiated and the torque on the motor is switched off.

Optional (dashed line) the window *SS1 velocity zero window* (n_Zero_SS1) can be monitored. If the speed is within the n_Zero_SS1 window for the *SS1 time for velocity zero* (t_L_SS1), the STO state can be activated immediately. The monitoring of the time t_L_SS1 is optional for this functionality.

Optionally the function SBC can be activated to close the brakes (see 8.3).

After activation of function SS1, it has to be finalized even if the request is reset before.

For the STO state the restart behaviour has to be taken into account (see 8.1.1).

The information SS1 state can be transferred in the standard process data. It signals that the function SS1 is activated.

8.2.1.2 SS1 with deceleration monitoring

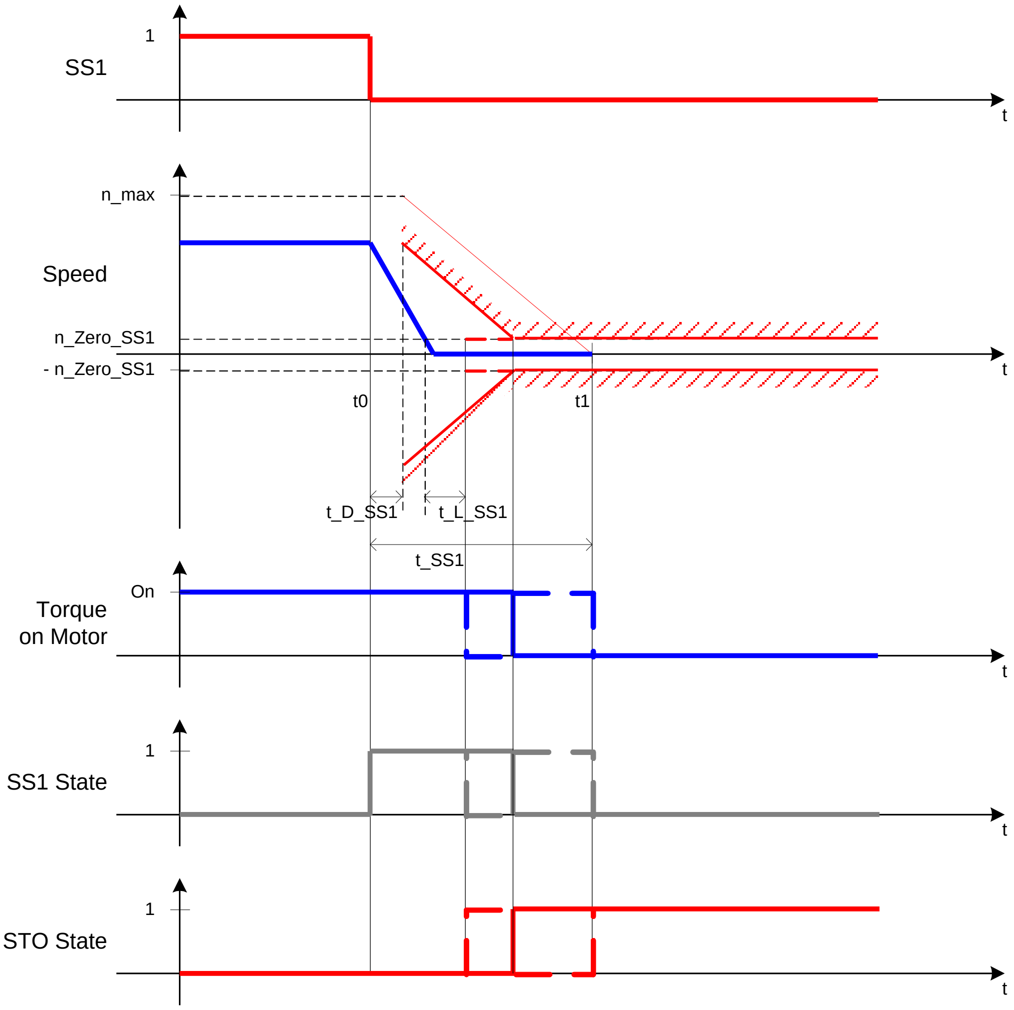

Figure 6: Functional principle of Safe Stop 1 (SS1) with deceleration limit

The same functionality as in "SS1 with time monitoring" (8.2.1.1) is used.

Additional functionality:

After activation of SS1 the *SS1 time delay deceleration monitoring* (t_D_SS1) is started. After this delay the monitoring of the deceleration of the speed is started.

The minimum deceleration ramp *SS1 deceleration limit* (a_SS1) is calculated as

|a_SS1| >= |n_max|/(t_SS1- t_D_SS1)

The delay time t_D_SS1 shall be lower than the time t_SS1. With activation of the SS1 function the current speed is latched. The speed monitoring starts with this speed value (i.e. the deceleration ramp is shifted parallel).

When the deceleration limit meets the n_Zero_SS1 limit, STO state can be initiated (this is latest after t_SS1 if the drive runs with maximum speed n_max).

Optionally the function SBC can be activated to close the brakes (see 8.3).

After activation of function SS1, it has to be finalized even if the request is reset before.

For the STO state the restart behaviour has to be taken into account (see 8.1.1).

The information SS1 state can be transferred in the standard process data. It signals that the function SS1 is activated.

8.2.2 Activation

Bit SS1 in the safety controlword.

8.2.3 Objects for SS1

The following objects are defined

Table 12: Object for SS1

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SS1 command	BOOLEAN	O	Read: 0: is not active 1: is active Write: 0: activate 1: deactivate
ARRAY (see 3.1.1)	SS1 time to STO	UNSIGNED16	M	Time delay to initiate STO Unit: [Time Units]
ARRAY (see 3.1.1)	SS1 velocity zero window u16	UNSIGNED16	C[1]	n=0 window Unit: [Velocity Units]
ARRAY (see 3.1.1)	SS1 velocity zero window u32	UNSIGNED32	C[1]	n=0 window Unit: [Velocity Units]
ARRAY (see 3.1.1)	SS1 time for velocity zero	UNSIGNED16	O	Time for speed within n=0 window Unit: [Time Units]
ARRAY (see 3.1.1)	SS1 deceleration limit u16	UNSIGNED16	C[2]	Deceleration limit for SS1 Unit: [Acceleration Units]
ARRAY (see 3.1.1)	SS1 deceleration limit u32	UNSIGNED32	C[2]	Deceleration limit for SS1 Unit: [Acceleration Units]
ARRAY (see 3.1.1)	SS1 time delay deceleration monitoring	UNSIGNED16	O	Time delay to activate deceleration limit Unit: [Time Units]
ARRAY (see 3.1.1)	SS1 activate SBC	UNSINGED32	O	Reference to SBC instance (*SBC command*) that should be activated 0x0000 0000 if no brake is used Format is fieldbus specific

[1] At least one of these objects is mandatory, if the safety function is supported

[2] At least one of these objects is mandatory, if the safety function is supported

8.2.4 Error reaction

Table 13: Error reaction SS1

Error reason	Behavior
SS1 velocity zero window not reached after *SS1 time to STO*	STO
SS1 deceleration limit is violated	STO

8.3 Safe Brake Control SBC

8.3.1 Description

8.3.1.1 SBC with STO

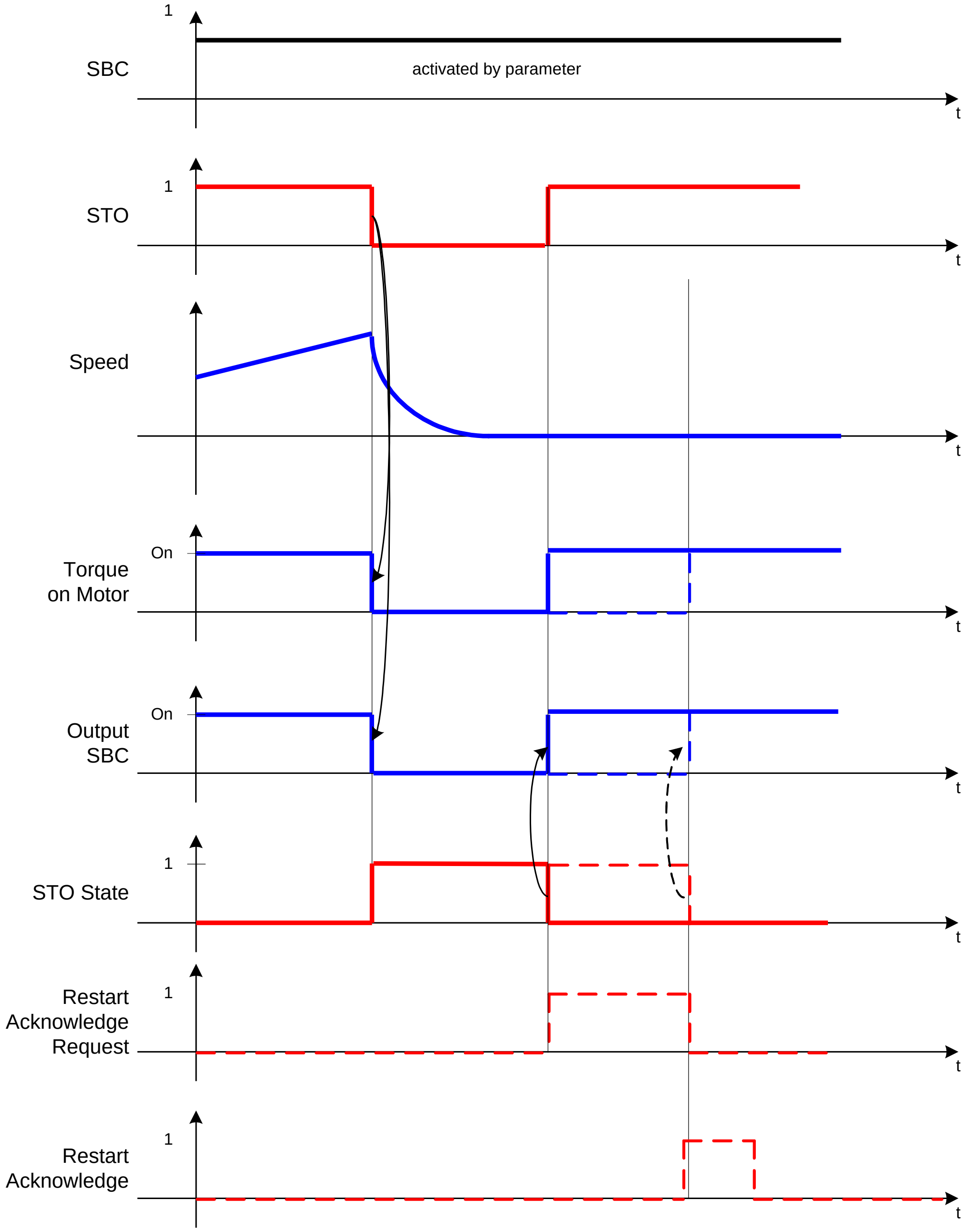

Figure 7: Functional principle of Safe Brake Control (SBC) with STO

The Output SBC can be implemented as a local output or as a bit in the manufacturer specific range of the safety statusword.

With activation of STO state or by a manufacturer specific bit in the safety controlword the output/bit SBC is set to Zero "0" so that the brakes will be closed. That means the brakes will be closed even if the motor is still running.

For the STO state the restart behaviour has to be taken into account (see 8.1.1).

The output/bit SBC is set to "1", if the STO state is deactivated and the optional manufacturer specific bit in the safety controlword is reset (="1").

8.3.1.2 SBC with SS1

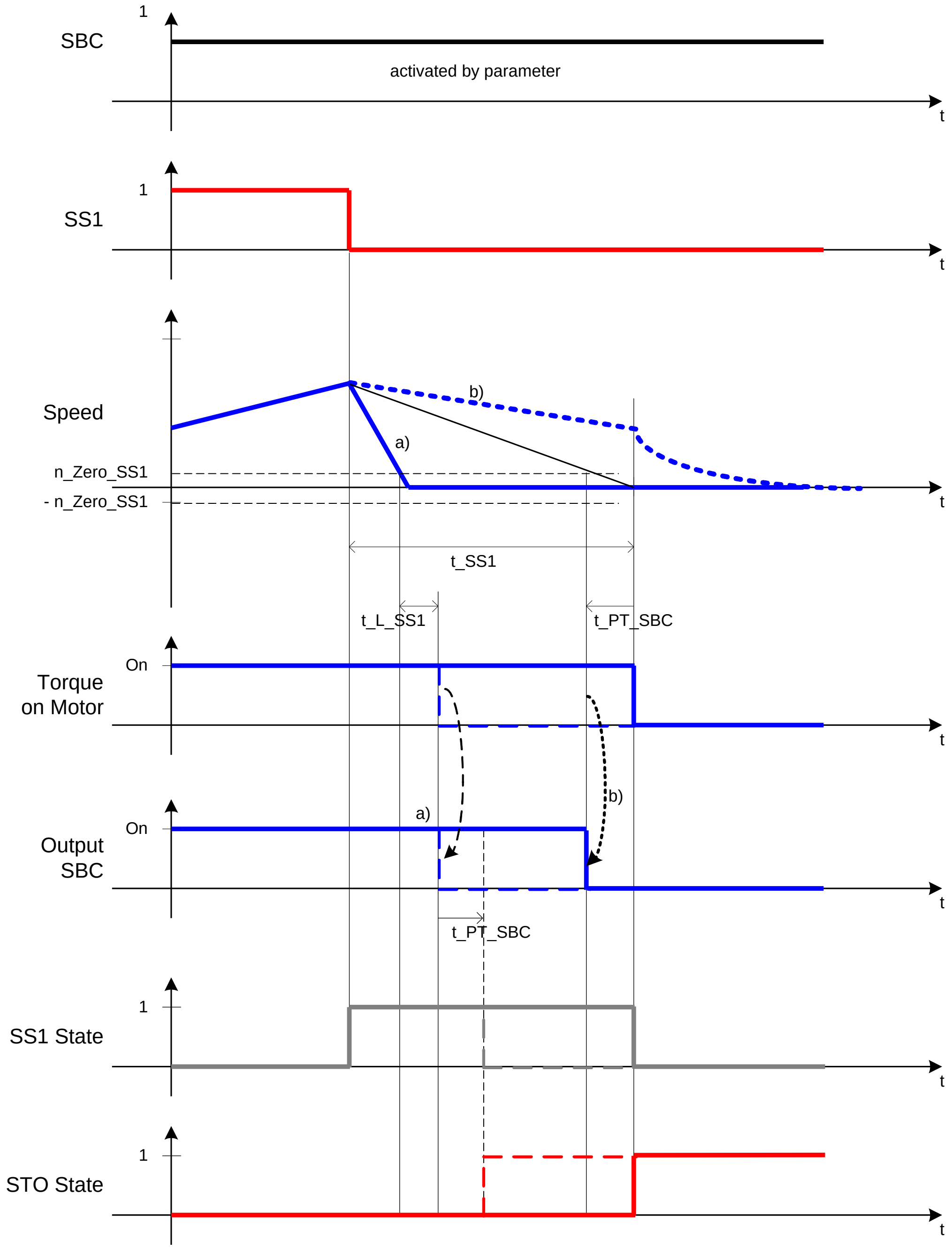

Figure 8: Functional principle of Safe Brake Control (SBC) with SS1

The *SBC brake time delay*(t_PT_SBC) is the time that is needed to close the brake.

If the optional *SS1 time for velocity zero*(t_L_SS1) is supported for SS1 and the speed is within the *SS1 velocity zero window* (n_Zero_SS1) for the time t_L_SS1, the output/bit SBC can be set to zero. The STO state can be activated after t_PT_SBC (a).

At least after the time *SS1 time to STO* (t_SS1) the STO state is activated. The output/bit SBC is set to zero at least at t_SS1-t_PT_SBC. That means that with activation of STO at t_SS1 the brake is closed (b).

The Output/bit SBC can also be set by a manufacturer specific bit in the safety controlword.

For the STO state the restart behaviour has to be taken into account (see 8.1.1).

The output/bit SBC is set to "1", if the STO state is deactivated and the optional manufacturer specific bit in the controlword is deactivated (corresponding bit in the controlword ="1").

8.3.2 Activation

SBC is activated by parameterization or by a manufacturer specific bit in the safety controlword.

8.3.3 Objects for SBC

The following objects are defined

Table 14: Objects for SBC

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SBC command	BOOLEAN	O	Read: 0: Brake Output 0, equals brake shall be closed 1: Brake Output 1, equals brake is open Write: 0: activate 1: deactivate
ARRAY (see 3.1.1)	SBC brake time delay	UNSIGNED16	O	Time needed to close the brake(s) Unit: [Time Units]

The instances of t_PT_SBC are needed if multiple brakes are supported.

8.3.4 Error reaction

No error reaction.

8.4 Safe Operating Stop SOS

8.4.1 Description

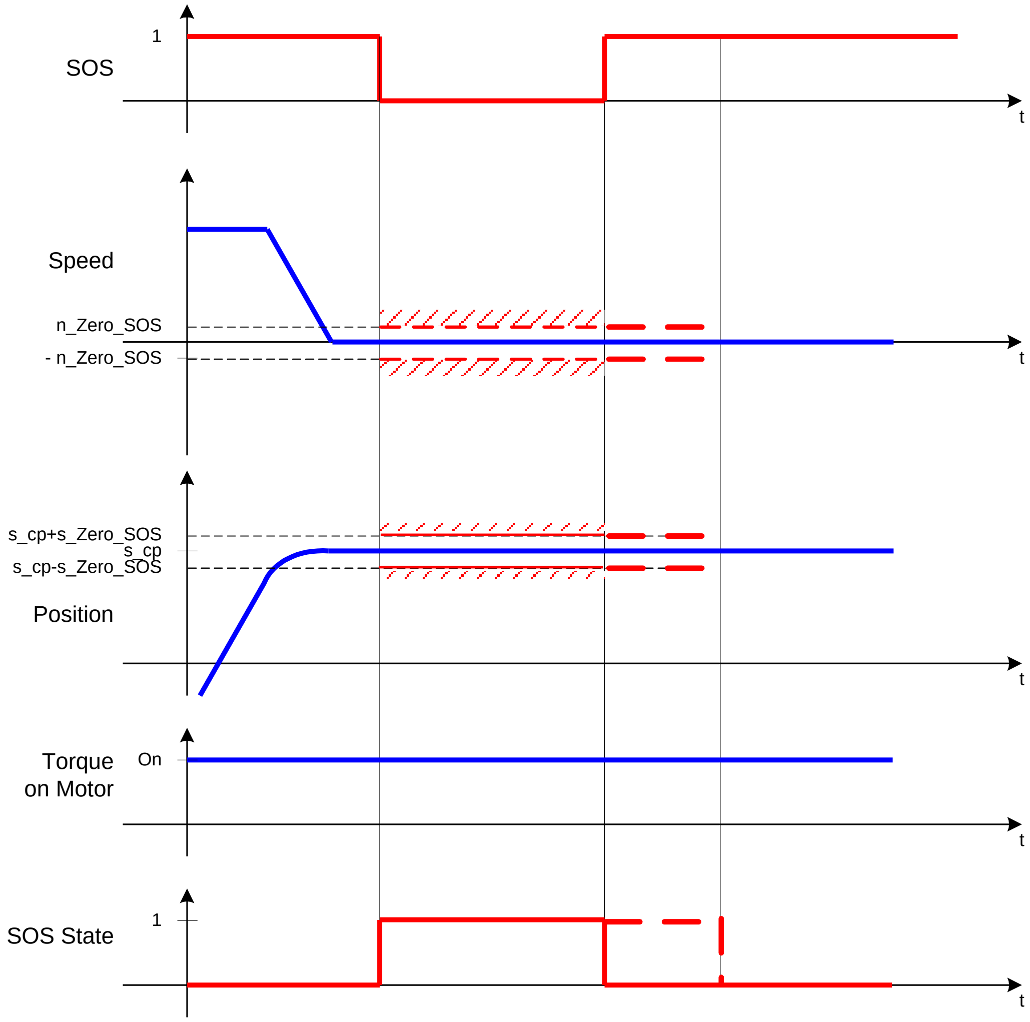

Figure 9: Functional principle of Safe Operating Stop (SOS)

With activation of SOS the monitoring of the current position s_cp is started. The position must be hold in the *SOS position zero window* (s_Zero_SOS); the limit s_cp ± s_Zero_SOS shall be observed. The PDS(SR) stays in control loop and retains the torque.

Optionally the speed can be monitored. The speed shall be hold in the limit *SOS velocity zero window* (± n_Zero_SOS).

8.4.2 Activation

Bit SOS in the safety controlword

8.4.3 Objects for SOS

The following objects are defined

Table 15: Objects for SOS

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SOS command	BOOLEAN	O	Read: 0: is not active 1: is active Write: 0: activate 1: deactivate

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SOS position zero window u16	UNSIGNED16	C[1]	Position window for stop position Unit: [Position Units]
ARRAY (see 3.1.1)	SOS position zero window u32	UNSIGNED32	C[1]	Position window for stop position Unit: [Position Units]
ARRAY (see 3.1.1)	SOS velocity zero window u16	UNSIGNED16	O	n=0 window Unit: [Velocity Units]
ARRAY (see 3.1.1)	SOS velocity zero window u32	UNSIGNED32	O	n=0 window Unit: [Velocity Units]

[1] At least one of these objects is mandatory, if the safety function is supported

8.4.4 Error reaction

Table 16: Error reaction SOS

Error reason	Behavior
SOS position zero window or *SOS velocity zero window* violated	STO

8.5 Safe Stop 2 SS2

8.5.1 Description

8.5.1.1 SS2 with time monitoring

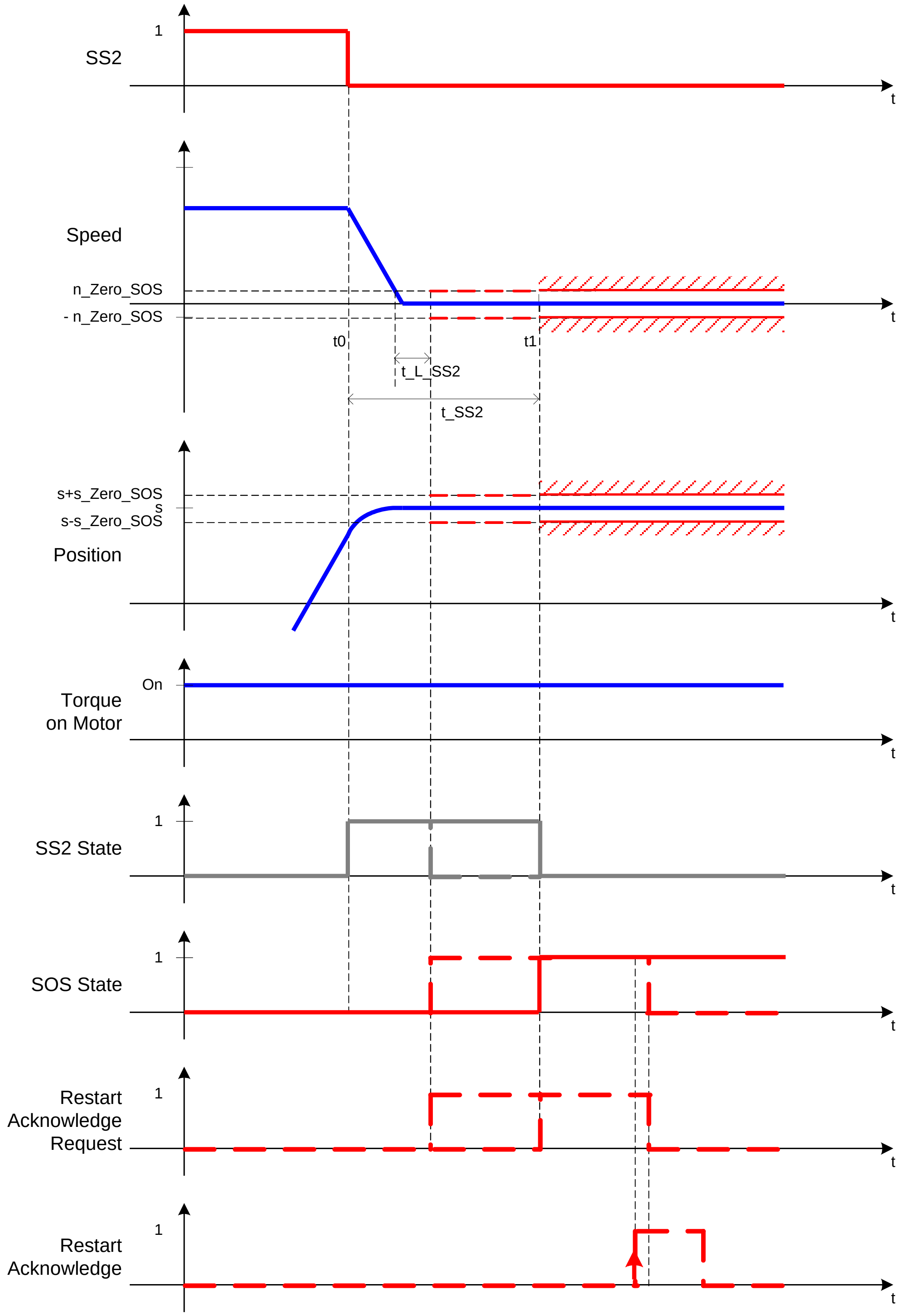

Figure 10: Functional principle of Safe Stop 2 (SS2) with time

With activation of SS2 the *SS2 time to SOS* (t_SS2) is started. The PDS(SR) starts with deceleration (standard task) and at least after t_SS2, the SOS state is initiated with the respective parameter.

NOTE: For each instance of the function SS2 the corresponding instance of SOS shall be used as the final state. I.e. for SS2_1 the SOS_1 function shall be used.

Optionally (dashed line) the *SOS velocity zero window* (n_Zero_SOS) can be monitored. If the speed is for the *SS2 time for velocity zero* (t_L_SS2) within the n_Zero_SS2 window, the SOS state can be activated immediately. The monitoring of the time t_L_SS2 is optional for this functionality.

After activation function SS2, it has to be finished even if the request is reset before.

For the final SOS state the restart behaviour has to be taken into account (see 7.1).

The information SS2 state can be transferred in the standard process data. It signals if the function SS2 is activated.

8.5.1.2 SS2 with deceleration monitoring

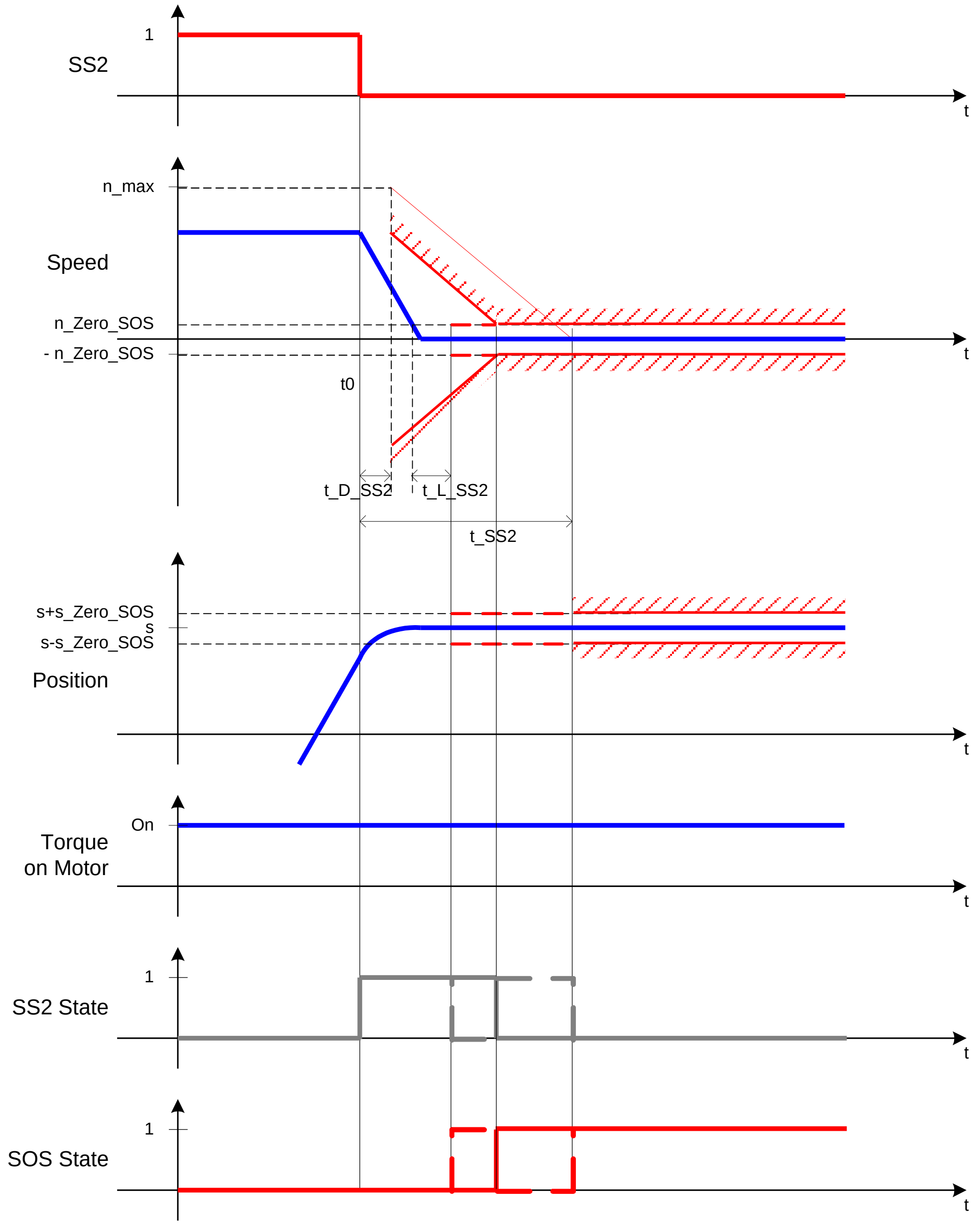

Figure 11: Functional principle of Safe Stop 2 (SS2) with deceleration

Same functionality as "SS2 with time monitoring" (8.5.1.1) is used.

Additional functionality:

After activation of SS2 the SS2 time delay deceleration monitoring (t_D_SS2) is started. After this delay the monitoring of the deceleration of the speed is started.

The minimum deceleration ramp *SS2 deceleration limit* (a_SS2) is calculated as

$$|a\_SS2| >= |n\_max|/(t\_SS2 - t\_D\_SS2)$$

The delay time t_D_SS2 shall be lower than the time t_SS2. With activation of the SS2 function the current speed is latched. The speed monitoring starts with this speed value (i.e. the deceleration ramp is shifted parallel).

When the deceleration limit meets the n_Zero_SOS limit the SOS state can be initiated (this is latest after t_SS2 if the drive runs with n_max).

After activation the function SS2 has to be finished even if the request is reset before.

For the SOS state the restart behaviour has to be taken into account (see 7.1).

The information SS2 state can be transferred in the standard process data. It signals if the function SS2 is activated.

8.5.2 Activation

Bit SS2 in the safety controlword

8.5.3 Objects for SS2

The following objects are defined

Table 17: Objects for SS2

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SS2 command	BOOLEAN	O	Read: 0: is not active 1: is active Write: 0: activate 1: deactivate
ARRAY (see 3.1.1)	SS2 time to SOS	UNSIGNED16	M	Time delay to initiate SOS Unit: [Time Units]
ARRAY (see 3.1.1)	SS2 time for velocity zero	UNSIGNED16	O	Time for speed within n=0 window Unit: [Time Units]
ARRAY (see 3.1.1)	SS2 deceleration limit u16	UNSIGNED16	O	Deceleration limit for SS2 Unit: [Acceleration Units]
ARRAY (see 3.1.1)	SS2 deceleration limit u32	UNSIGNED32	O	Deceleration limit for SS2 Unit: [Acceleration Units]
ARRAY (see 3.1.1)	SS2 time delay deceleration monitoring	UNSIGNED16	O	Time delay to activate deceleration limit Unit: [Time Units]
ARRAY (see 3.1.1)	SS2 restart acknowledge behavior	BOOLEAN	O	0: Restart Acknowledge for SS2 switched-off (automatic restart) 1: Restart Acknowledge for SS2 switched on
ARRAY (see 3.1.1)	SS2 error reaction	UNSIGNED32	O	Error reaction for violation of a_SS2 Default: STO No Reaction: 0x0000 0000

8.5.4 Error reaction

Table 18: Error reaction SS2

Error reason	Behavior
SS2 deceleration limit violated	see object *SS2 error reaction*
SOS violated	*SOS error reaction*

8.6 Safe Speed Range SSR

8.6.1 Description

8.6.1.1 SSR with time monitoring

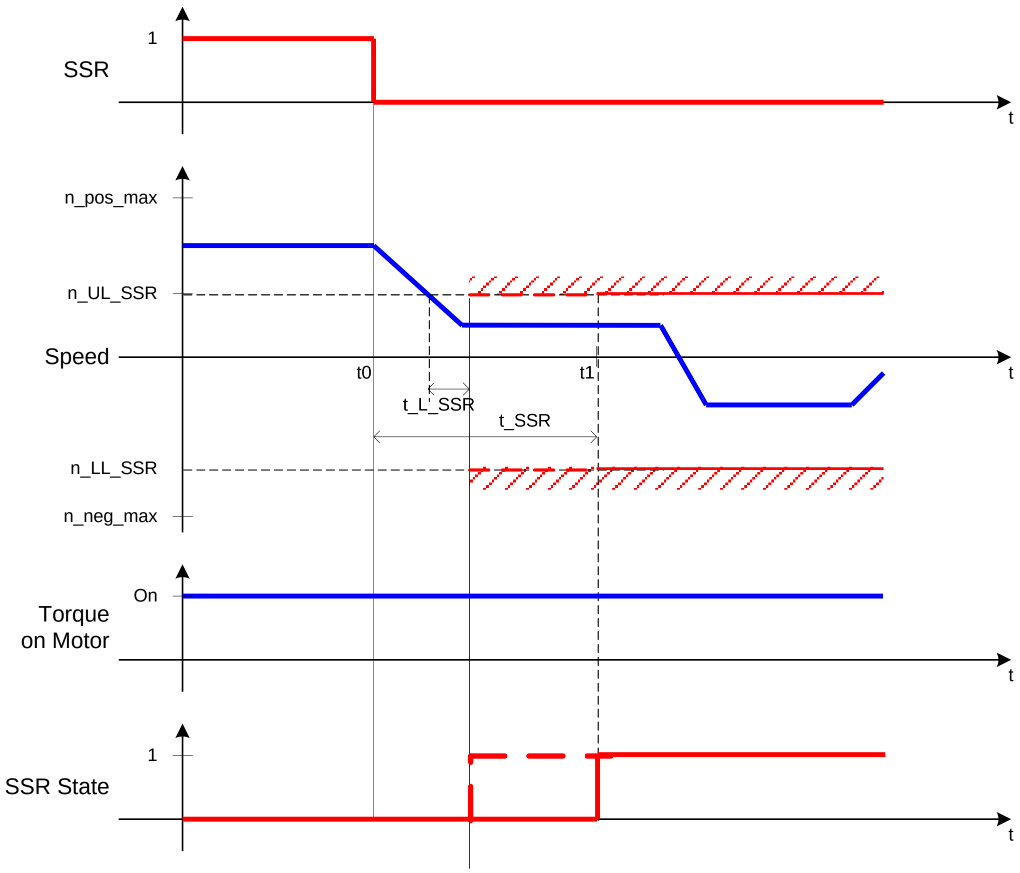

Figure 12: Functional principle of Safe Speed Range (SSR) with Time limit

With activation of SSR the *SSR time to velocity monitoring* (t_SSR) is started. The PDS(SR) starts with deceleration (standard task) and at least after t_SSR, the monitoring of the SSR velocity limits n_UL_SSR (upper limit) and n_LL_SSR (lower limit) is active.

Optionally (dashed line) the monitoring can start immediately if the speed is SSR time for velocity in limits (t_L_SSR) within the speed limits..

Usually n_UL_SSR is a positive value and n_LL_SSR is a negative value.

8.6.1.2 SSR with deceleration monitoring

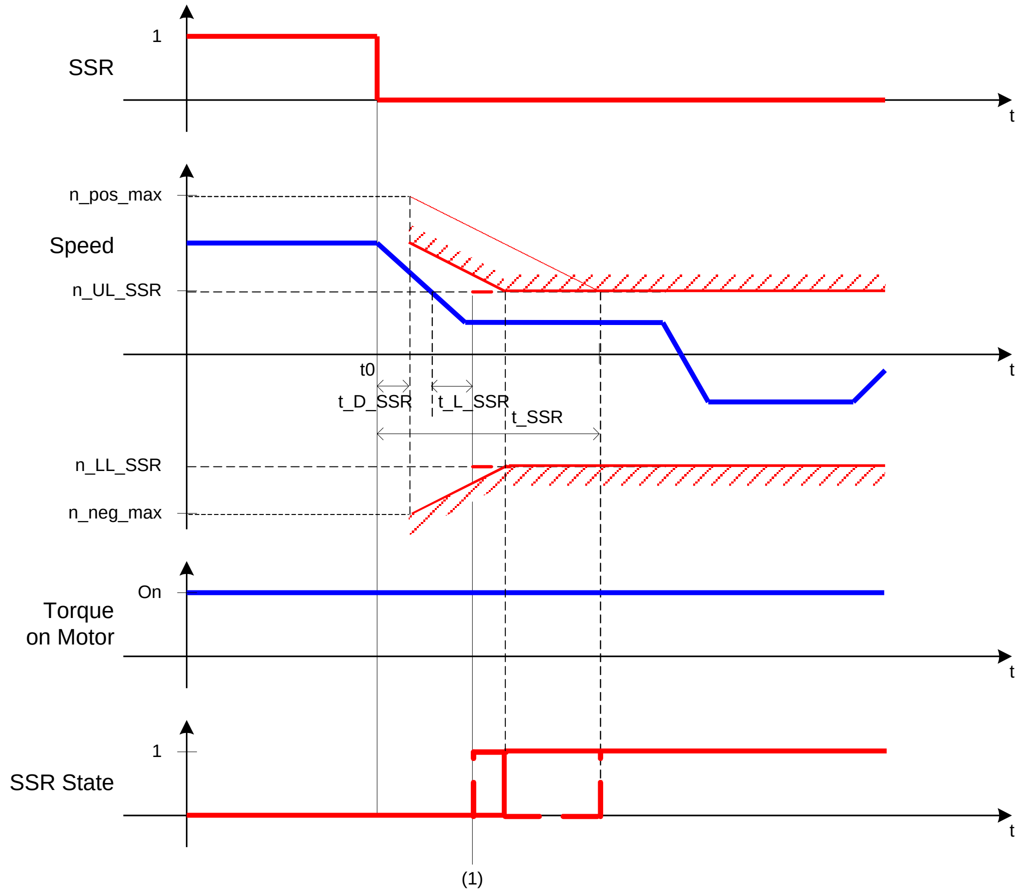

Figure 13: Functional principle of Safe Speed Range (SSR) with Deceleration limit

Same functionality as "SSR with time monitoring" (8.6.1.1) is used.

Additional functionality:

After activation of SSR the *SSR time delay deceleration monitoring* (t_D_SSR) is started. After this delay the monitoring of the deceleration of the speed is started.

The minimum deceleration ramp *SSR deceleration limit* is calculated as

|a_SSR| >= min{ | n_pos_max – n_UL_SSR|/(t_SSR- t_D_SSR);
| n_neg_max + n_LL_SSR|/(t_SSR- t_D_SSR)
}

The delay time t_D_SSR shall be lower than the time t_SSR. With activation of the SSR function the current speed is latched. The speed monitoring starts with this speed value (i.e. the deceleration ramp is shifted parallel).

When the deceleration limit meets the n_UL_SSR limit and the n_LL_SSR limit the SSR state can be initiated (this is latest after t_SSR if the drive runs with n_max).

8.6.2 Activation

Bit SSR in the safety controlword

8.6.3 Objects for SSR

The following objects are defined

Table 19: Objects for SSR

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SSR command	BOOLEAN	O	Read: 0: is not active 1: is active Write: 0: activate 1: deactivate
ARRAY (see 3.1.1)	SSR time to velocity monitoring	UNSIGNED16	O	Time delay to initiate SSR limits Unit: [Time Units]
ARRAY (see 3.1.1)	SSR velocity upper limit i16	INTEGER16	C[1]	Upper velocity limit Unit: [Velocity Units]
ARRAY (see 3.1.1)	SSR velocity upper limit i32	INTEGER32	C[1]	Upper velocity limit Unit: [Velocity Units]
ARRAY (see 3.1.1)	SSR velocity lower limit i16	INTEGER16	C[2]	Lower velocity limit Unit: [Velocity Units]
ARRAY (see 3.1.1)	SSR velocity lower limit i32	INTEGER32	C[2]	Lower velocity limit Unit: [Velocity Units]
ARRAY (see 3.1.1)	SSR time for velocity in limits	UNSIGNED16	O	Time for speed within n_UL_SSR and n_LL_SSR window Unit: [Time Units]
ARRAY (see 3.1.1)	SSR deceleration limit u16	UNSIGNED16	O	Deceleration limit for SSR Unit: [Acceleration Units]
ARRAY (see 3.1.1)	SSR deceleration limit u16	UNSIGNED32	O	Deceleration limit for SSR Unit: [Acceleration Units]
ARRAY (see 3.1.1)	SSR time delay deceleration monitoring	UNSIGNED16	O	Time delay to activate deceleration limit Unit: [Time untis]
ARRAY (see 3.1.1)	SSR error reaction	UNSIGNED32	O	Error reaction for violation of n_UL_SSR, n_LL_SSR or a_SSR limits Default: STO No Reaction: 0x0000 0000

[1] At least one of these objects is mandatory, if the safety function is supported

[2] At least one of these objects is mandatory, if the safety function is supported

8.6.4 Error reaction

Table 20: Error reaction SSR

Error reason	Behavior
SSR velocity upper/lower limit violated	see object *SSR error reaction*
SSR deceleration limit violated	see object *SSR error reaction*

Note : The time delays of the stop function (SS1, SS2) are added to the error reaction time.

8.7 Safe Acceleration Range SAR

8.7.1 Description

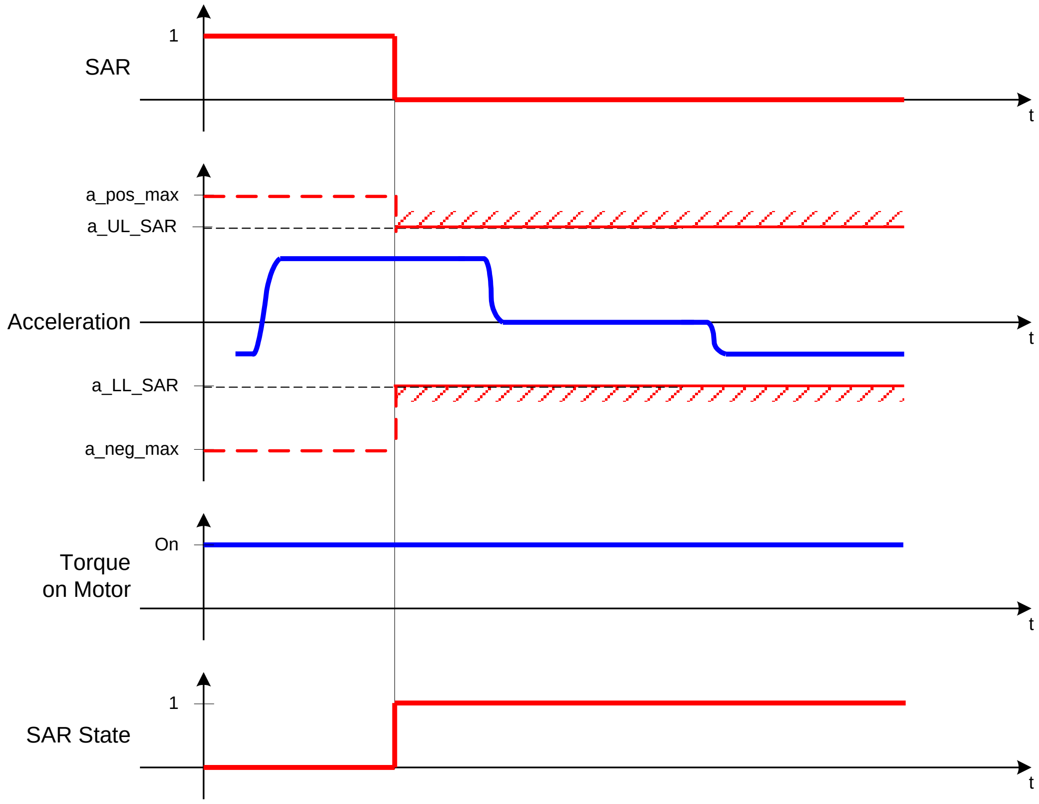

Figure 14: Functional principle of Safe Acceleration Range (SAR)

With activation of SAR the monitoring of the acceleration window is started. The acceleration must be hold in the *SAR acceleration upper limit* (a_UL_SAR) and *SAR acceleration lower limit* (a_LL_SAR).

Optionally the maximum acceleration can be monitored, see 8.11.

8.7.2 Activation

Bit SAR in the safety controlword

8.7.3 Objects for SAR

The following objects are defined

Table 21: Objects for SAR

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SAR command	BOOLEAN	O	Read: 0: is not active 1: is active Write: 0: activate 1: deactivate
ARRAY (see 3.1.1)	SAR acceleration upper limit i16	INTEGER16	C[1]	Upper acceleration limit Unit: [Acceleration Units]
ARRAY (see 3.1.1)	SAR acceleration upper limit i32	INTEGER32	C[1]	Upper acceleration limit Unit: [Acceleration Units]
ARRAY (see 3.1.1)	SAR acceleration lower limit i16	INTEGER16	C[2]	Lower acceleration limit Unit: [Acceleration Units]

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SAR acceleration lower limit i32	INTEGER32	C[2]	Lower acceleration limit Unit: [Acceleration Units]
ARRAY (see 3.1.1)	SAR error reaction	UNSIGNED32	O	Error reaction for violation of n_UL_SAR, n_LL_SAR limits Default: STO No Reaction: 0x0000 0000

[1] At least one of these objects is mandatory, if the safety function is supported

[2] At least one of these objects is mandatory, if the safety function is supported

8.7.4 Error reaction

Table 22: Error reaction SAR

Error reason	Behavior
SAR acceleration upper limit or *SAR acceleration lower limit* i16 violated	see object *SAR error reaction*

8.8 Safely-limited Speed SLS

8.8.1 Description

8.8.1.1 SLS with time monitoring

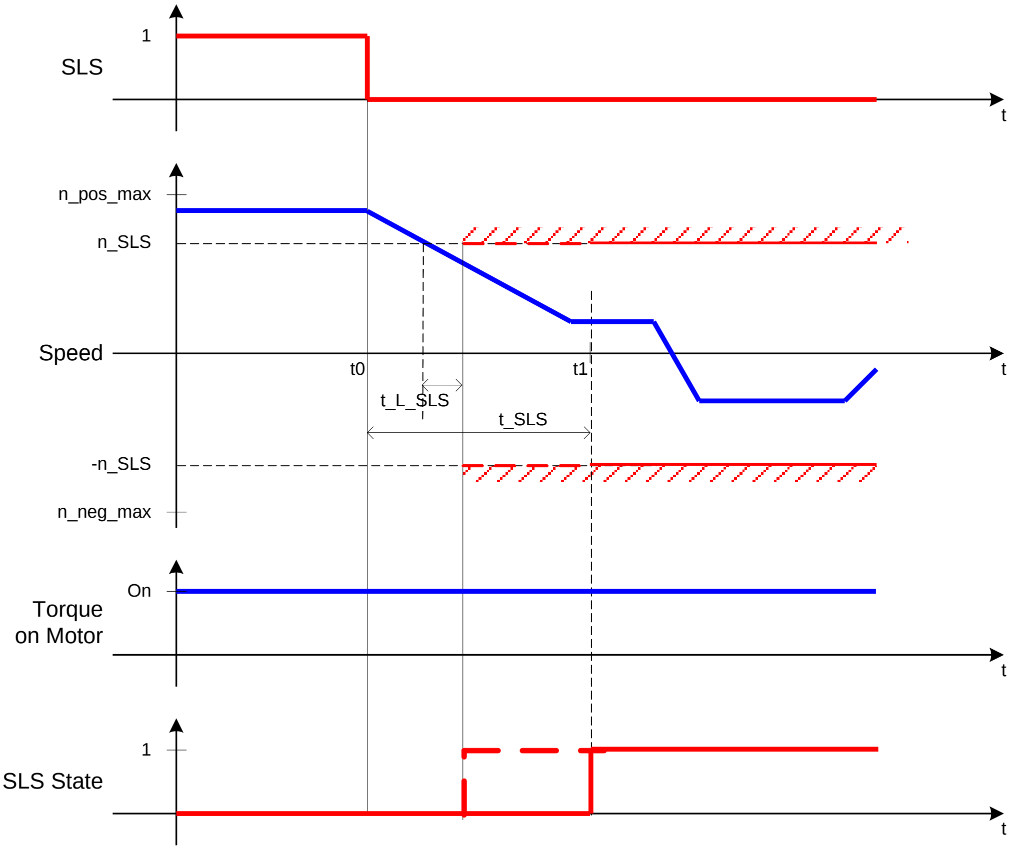

Figure 15: Functional principle of Safely-limited Speed (SLS) with time

With activation of SLS the *SLS time to velocity monitoring* (t_SLS) is started. The PDS(SR) starts with deceleration (standard task) and at least after t_SLS, the monitoring of the *SLS velocity limit* (± n_SLS) is active.

Optionally (dashed line) the monitoring can start immediately if the speed is for *SLS time for velocity in limits* (t_L_SLS) within the speed limits.

8.8.1.2 SLS with deceleration monitoring

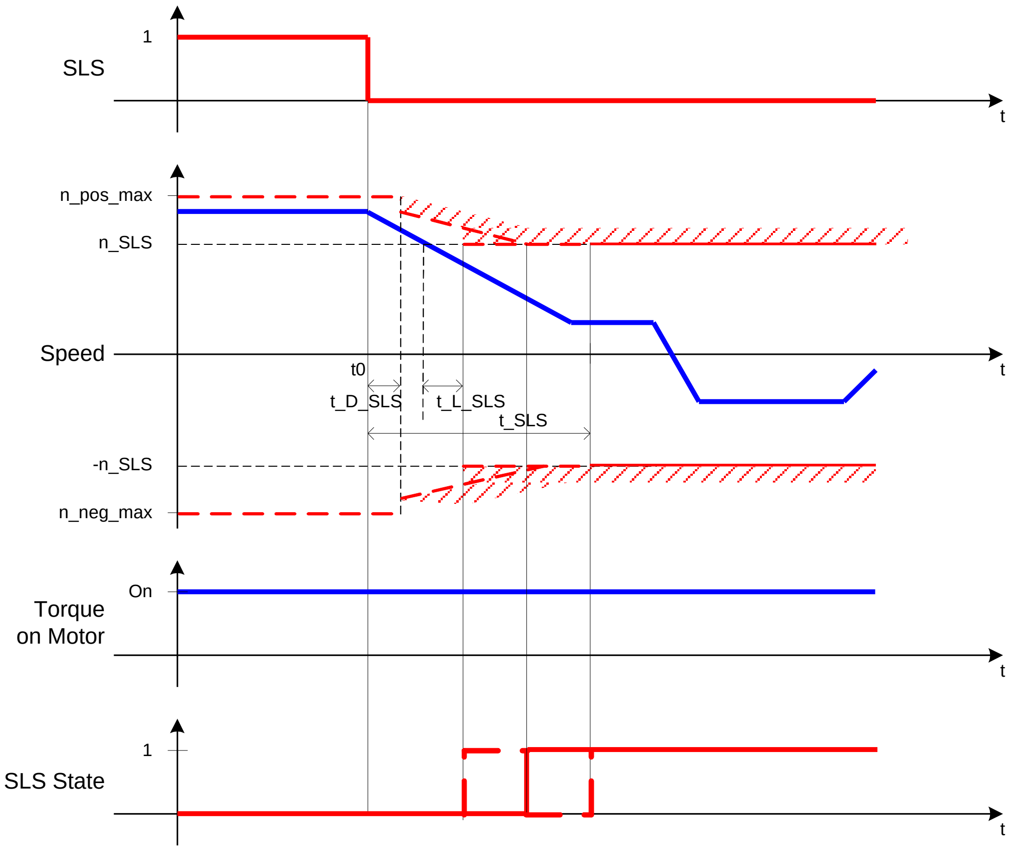

Figure 16: Functional principle of Safely-limited Speed (SLS) with deceleration

Same functionality as "SLS with time monitoring" (8.8.1.1) is used.

Additional functionality:

After activation of SLS the *SLS time delay deceleration monitoring* (t_D_SLS) is started. After this delay the monitoring of the deceleration of the speed is started.

The minimum *SLS deceleration limit* ramp (a_SLS) is calculated as

$$|a\_SLS| >= |n\_max - n\_SLS|/(t\_SLS - t\_D\_SLS)$$

The delay time t_D_SLS shall be lower than the time t_SLS. With activation of the SLS function the current speed is latched. The speed monitoring starts with this speed value (i.e. the deceleration ramp is shifted parallel).

When the deceleration limit meets the *SLS velocity limit* (n_SLS) the SLS state can be initiated (this is latest after t_SLS if the drive runs with n_max).

8.8.2 Activation

Bit SLS in the safety controlword

8.8.3 Objects for SLS

The following objects are defined

Table 23: Objects for SLS

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SLS command	BOOLEAN	O	Read: 0: is not active 1: is active Write: 0: activate 1: deactivate
ARRAY (see 3.1.1)	SLS time to velocity monitoring	UNSIGNED16	O	Time delay to initiate SLS limits Unit: [Time Units]
ARRAY (see 3.1.1)	SLS velocity limit u16	UNSIGNED16	C[1]	Velocity limit Unit: [Velocity Units]
ARRAY (see 3.1.1)	SLS velocity limit u32	UNSIGNED32	C[1]	Velocity limit Unit: [Velocity Units]
ARRAY (see 3.1.1)	SLS time for velocity in limits	UNSIGNED16	O	Time for speed within n_SLS window Unit: [Time Units]
ARRAY (see 3.1.1)	SLS deceleration limit u16	UNSIGNED16	O	Deceleration limit for SLS Unit: [Acceleration Units]
ARRAY (see 3.1.1)	SLS deceleration limit u32	UNSIGNED32	O	Deceleration limit for SLS Unit: [Acceleration Units]
ARRAY (see 3.1.1)	SLS time delay deceleration monitoring	UNSIGNED16	O	Time delay to activate deceleration limit Unit: [Time untis]
ARRAY (see 3.1.1)	SLS error reaction	UNSIGNED32	O	Error reaction for violation of n_SLS or a_SLS limits Default: STO No Reaction: 0x0000 0000

[1] At least one of these objects is mandatory, if the safety function is supported

8.8.4 Error Reaction

Table 24: Error Reaction SLS

Error Reason	Behaviour
SLS velocity limit violated	see object *SLS error reaction*
SLS deceleration limit violated	see object *SLS error reaction*

Note: The time delays of the stop function (SS1, SS2) are added to the error reaction time.

8.9 Safely-limited Position SLP

8.9.1 Description

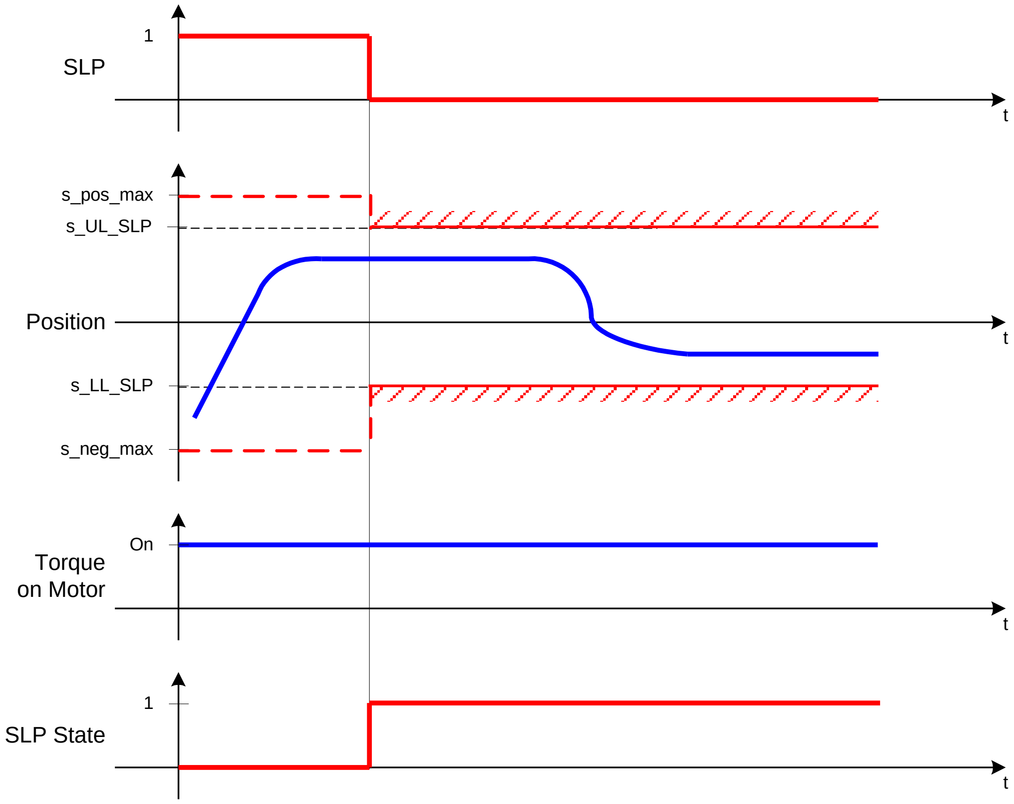

Figure 17: Functional principle of Safely-limited Position (SLP)

With activation of SLP the monitoring of the *SLP position upper limit* (s_UL_SLP) and *SLP position lower limit* (s_LL_SLP) are active.

s_LL_SLP <= pos <= s_UL_SLP

8.9.2 Activation

Bit SLP in the safety controlword

8.9.3 Objects for SLP

The following objects are defined

Table 25: Objects for SLP

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SLP command	BOOLEAN	O	Read: 0: is not active 1: is active Write: 0: activate 1: deactivate
ARRAY (see 3.1.1)	SLP position upper limit i16	INTEGER16	C^1	Upper position limit Unit: [Position Units]
ARRAY (see 3.1.1)	SLP position upper limit i32	INTEGER32	C^1	Upper position limit Unit: [Position Units]
ARRAY (see 3.1.1)	SLP position lower limit i16	INTEGER16	C^2	Lower position limit Unit: [Position Units]

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SLP position lower limit i32	INTEGER32	C[2]	Lower position limit Unit: [Position Units]
ARRAY (see 3.1.1)	SLP error reaction	UNSIGNED32	O	Error reaction for violation of n_UL_SLP or n_LL_SLP limits Default: STO No Reaction: 0x0000 0000

[1] At least one of these objects is mandatory, if the safety function is supported

[2] At least one of these objects is mandatory, if the safety function is supported

8.9.4 Error reaction

Table 26: Error reaction SLP

Error Reason	Behaviour
SLP position upper limit or *SLP position lower limit* violated	see object *SLP error reaction*

8.10 Safe Maximum Speed SMS

8.10.1 Description

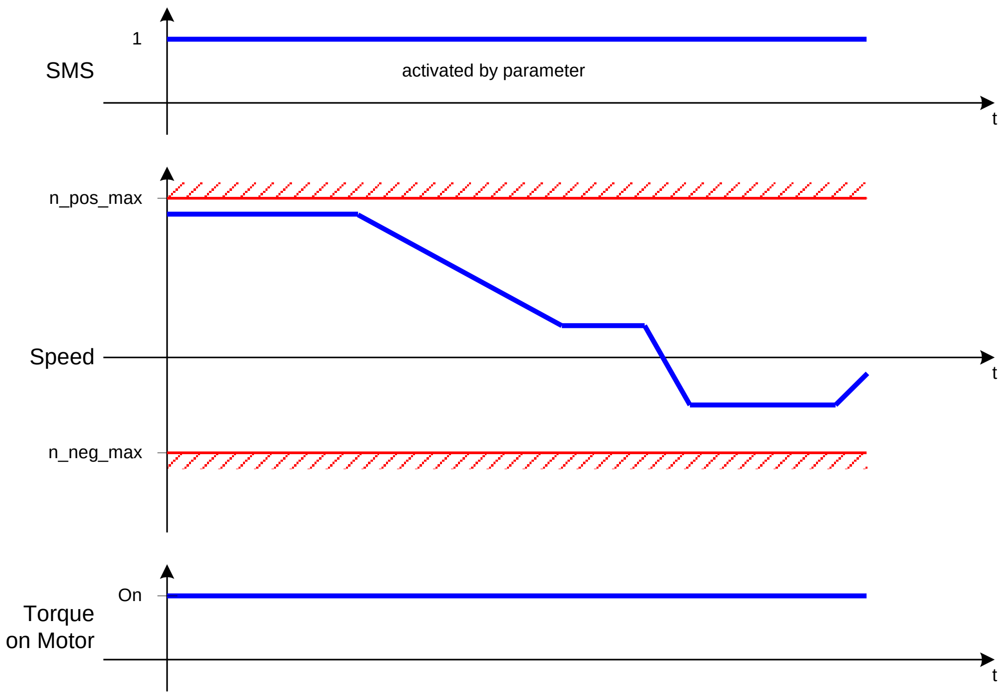

Figure 18: Functional principle of Safe Maximum Speed (SMS)

The maximum speed is monitored within the ranges *SMS velocity maximum positive* (n_pos_max) and *SMS velocity maximum negative* (n_neg_max).

This function is activated if the objects n_pos_max and n_neg_max are supported and at least one of the objects is not Zero.

8.10.2 Activation

By parameterization

If objects n_pos_max and n_neg_max are supported and at least one is unequal zero SMS is activated.

8.10.3 Objects for SMS

The following objects are defined

Table 27: Objects for SMS

Object Code	Name	Type	M/O/C	Value
VAR	SMS status	BOOLEAN	O	Read: 0: is not active 1: is active
VAR	SMS velocity maximum positive u16	INTEGER16	C[1]	Maximum velocity limit positive direction Unit: [Velocity Units]
VAR	SMS velocity maximum positive u32	INTEGER32	C[1]	Maximum velocity limit positive direction Unit: [Velocity Units]
VAR	SMS velocity maximum negative u16	INTEGER16	C[2]	Maximum velocity limit negative direction Unit: [Velocity Units]
VAR	SMS velocity maximum negative u32	INTEGER32	C[2]	Maximum velocity limit negative direction Unit: [Velocity Units]
VAR	SMS error reaction	UNSIGNED32	O	Error reaction for violation of n_pos_max or n_neg_max limits Default: STO No Reaction: 0x0000 0000

[1] At least one of these objects is mandatory, if the safety function is supported

[2] At least one of these objects is mandatory, if the safety function is supported

The speed n_max (used in other functions) is the maximum of the absolute values of n_pos_max and n_neg_max.

8.10.4 Error reaction

Table 28: Error reaction SMS

Error Reason	Behaviour
SMS maximum positive velocity or *SMS maximum negative velocity* violated	see object *SMS error reaction*

8.11 Safely-limited Torque SLT

8.11.1 Description

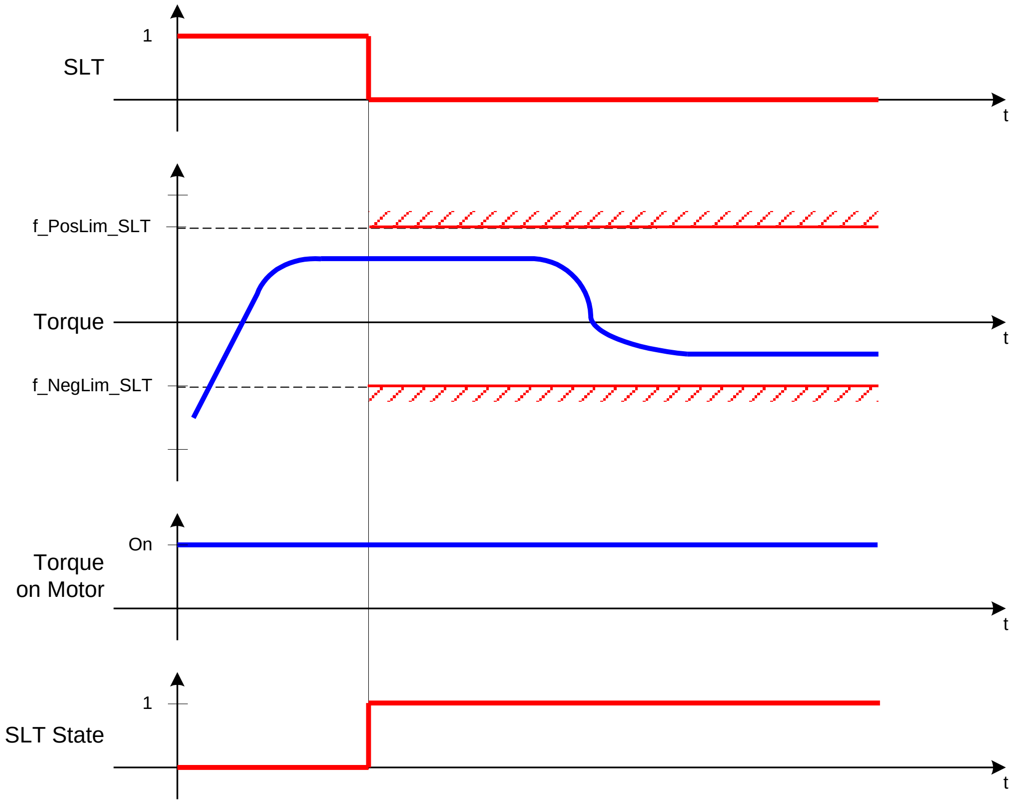

Figure 19: Functional principle of Safety-limited Torque (SLT)

With activation of SLT the monitoring of the *SLT torque limit positive direction* (f_PosLim_SLT) and *SLT torque limit negative direction* (f_NegLim_SLT) are active.

8.11.2 Activation

Bit SLT in the safety controlword

8.11.3 Objects for SLT

The following objects are defined

Table 29: Objects SLT supported

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SLT command	BOOLEAN	O	Read: 0: is not active 1: is active Write: 0: activate 1: deactivate
ARRAY (see 3.1.1)	SLT torque limit positive direction	UNSIGNED16	C[1]	Torque limit for positive direction Unit: [Torque Units]
ARRAY (see 3.1.1)	SLT torque limit negative direction	UNSIGNED16	C[1]	Torque limit for negative direction Unit: [Torque Units]

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SLT error reaction	UNSIGNED32	O	Error reaction for violation of f_PosLim_SLT or f_NegLim_SLT limits Default: STO No Reaction: 0x0000 0000

[1] At least one of these objects is mandatory, if the safety function is supported

8.11.4 Error reaction

Table 30: Error reaction SLT

Error reason	Behavior
SLT torque limit positive direction or *SLT torque limit negative direction* violated	see object *SLT error reaction*

8.12 Safe Maximum Acceleration SMA

8.12.1 Description

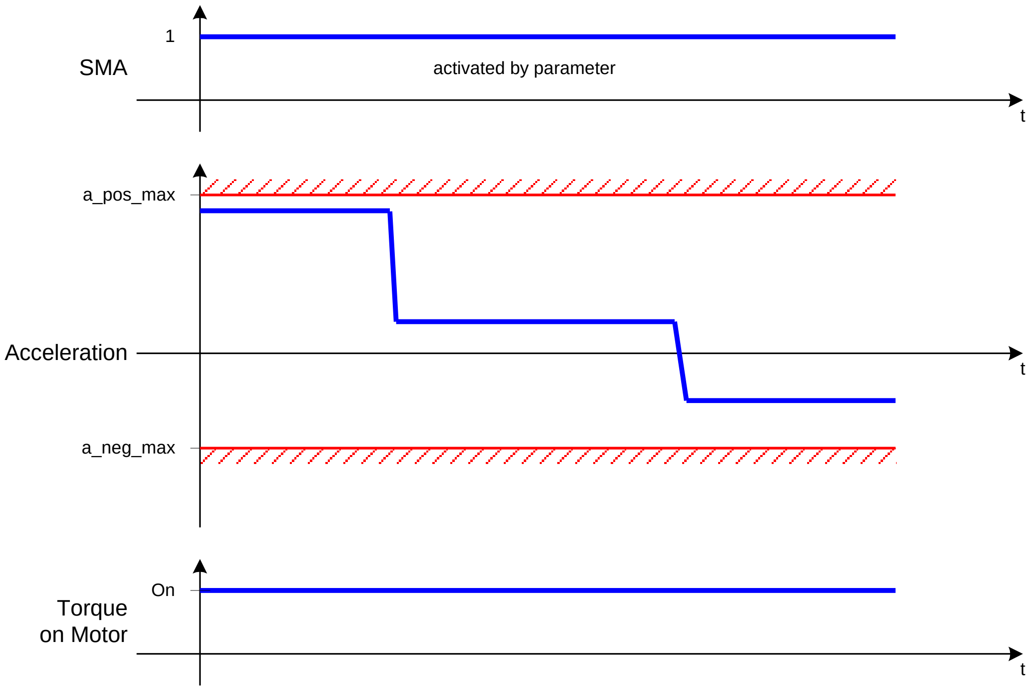

Figure 20: Functional principle of Safe Maximum Acceleration (SMA)

The maximum acceleration is monitored. This function is activated if the objects *SMA acceleration maximum* (a_pos_max) and *SMA deceleration maximum (*a_neg_max) are support and at least one of the objects is not zero.

8.12.2 Activation

By parametrization

If the objects a_pos_max and a_neg_max are supported and at least one is unequal zero SMA is activated.

8.12.3 Objects for SMA

The following objects are defined

Table 31: Objects for SMA

Object Code	Name	Type	M/O/C	Value
VAR	SMA status	BOOLEAN	O	Read: 0: is not active 1: is active
VAR	SMA acceleration maximum i16	INTEGER16	C[1]	Upper maximum acceleration limit Unit: [Acceleration Units]
VAR	SMA acceleration maximum i32	INTEGER32	C[1]	Upper maximum acceleration limit Unit: [Acceleration Units]
VAR	SMA deceleration maximum i16	INTEGER16	C[2]	Lower maximum acceleration limit Unit: [Acceleration Units]
VAR	SMA deceleration maximum i32	INTEGER32	C[2]	Lower maximum acceleration limit Unit: [Acceleration Units]
VAR	SMA error reaction	UNSIGNED32	O	Error reaction for violation of a_pos_max or a_neg_max limits Default: STO No Reaction: 0x0000 0000

[1] At least one of these objects is mandatory, if the safety function is supported

[2] At least one of these objects is mandatory, if the safety function is supported

8.12.4 Error reaction

Table 32: Error reaction SMA

Error reason	Behavior
SMA maximum acceleration or *SMA maximum deceleration* violated	see object *SMA error reaction*

8.13 Safely-Limited Increment SLI

8.13.1 Description

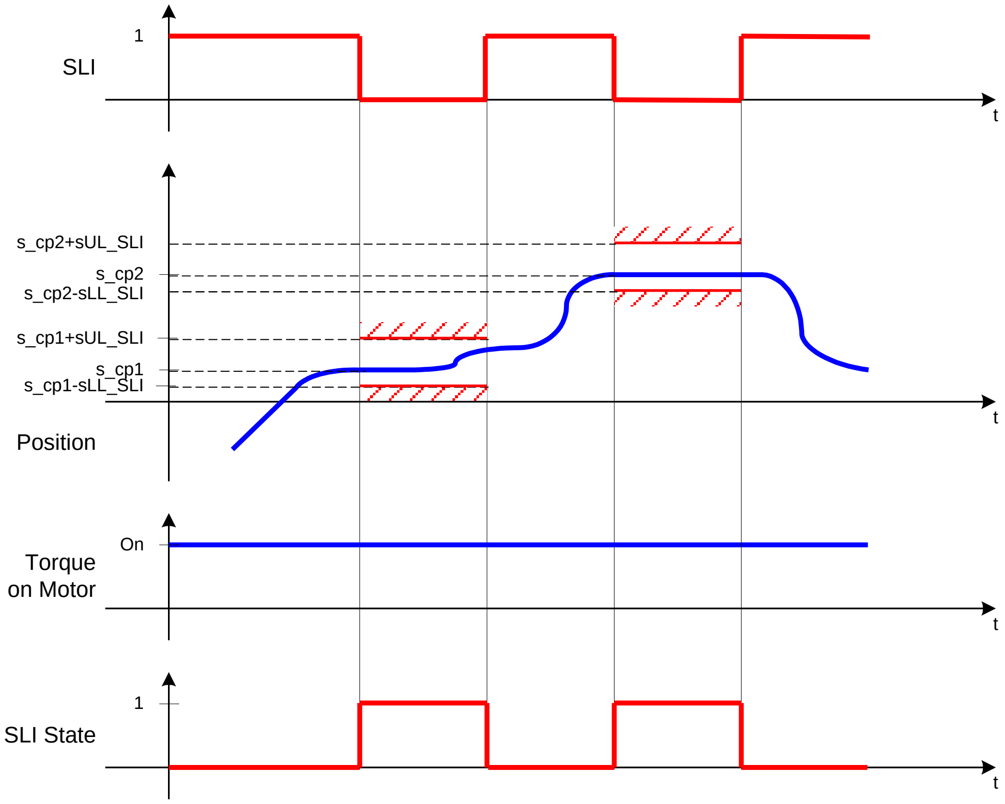

Figure 21: Functional principle of Safely-Limited increment (SLI)

If the bit SLI is set to zero "0" the monitoring of SLI is activated.

With activation of SLI the current position s is latched (s_cp) and the monitoring is started. The position must be hold in the *SLI position relative upper limit* s_cp+s_UL_SLI and *SLI position relative lower limit* s_cp-s_LL_SLI window. The PDS(SR) stays in control loop and moves only a limited number of increments.

There is no monitoring of the restart behaviour assigned to this function.

8.13.2 Activation

Bit SLI in the safety controlword

8.13.3 Objects for SLI

The following objects are defined

Table 33: Objects for SLI

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SLI command	BOOLEAN	O	Read: 0: is not active 1: is active Write: 0: activate 1: deactivate
ARRAY (see 3.1.1)	SLI position relative upper limit i16	INTEGER16	C[1]	Upper relative position limit Unit: [Position Units]
ARRAY (see 3.1.1)	SLI position relative upper limit i32	INTEGER32	C[1]	Upper relative position limit Unit: [Position Units]

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SLI position relative lower limit i16	INTEGER16	C[2]	Lower relativ position limit Unit: [Position Units]
ARRAY (see 3.1.1)	SLI position relative lower limit i32	INTEGER32	C[2]	Lower relativ position limit Unit: [Position Units]
ARRAY (see 3.1.1)	SLI error reaction	UNSIGNED32	O	Error reaction for violation of s_UL_SLI, s_LL_SLI limits Default: STO No Reaction: 0x0000 0000

[1] At least one of these objects is mandatory, if the safety function is supported

[2] At least one of these objects is mandatory, if the safety function is supported

8.13.4 Error reaction

Table 34: Error reaction SLI

Error reason	Behavior
SLI relative position upper limit or *SLI relative position lower limit* violated	see object *SLI error reaction*

8.14 Safe Direction SDI

8.14.1 Description

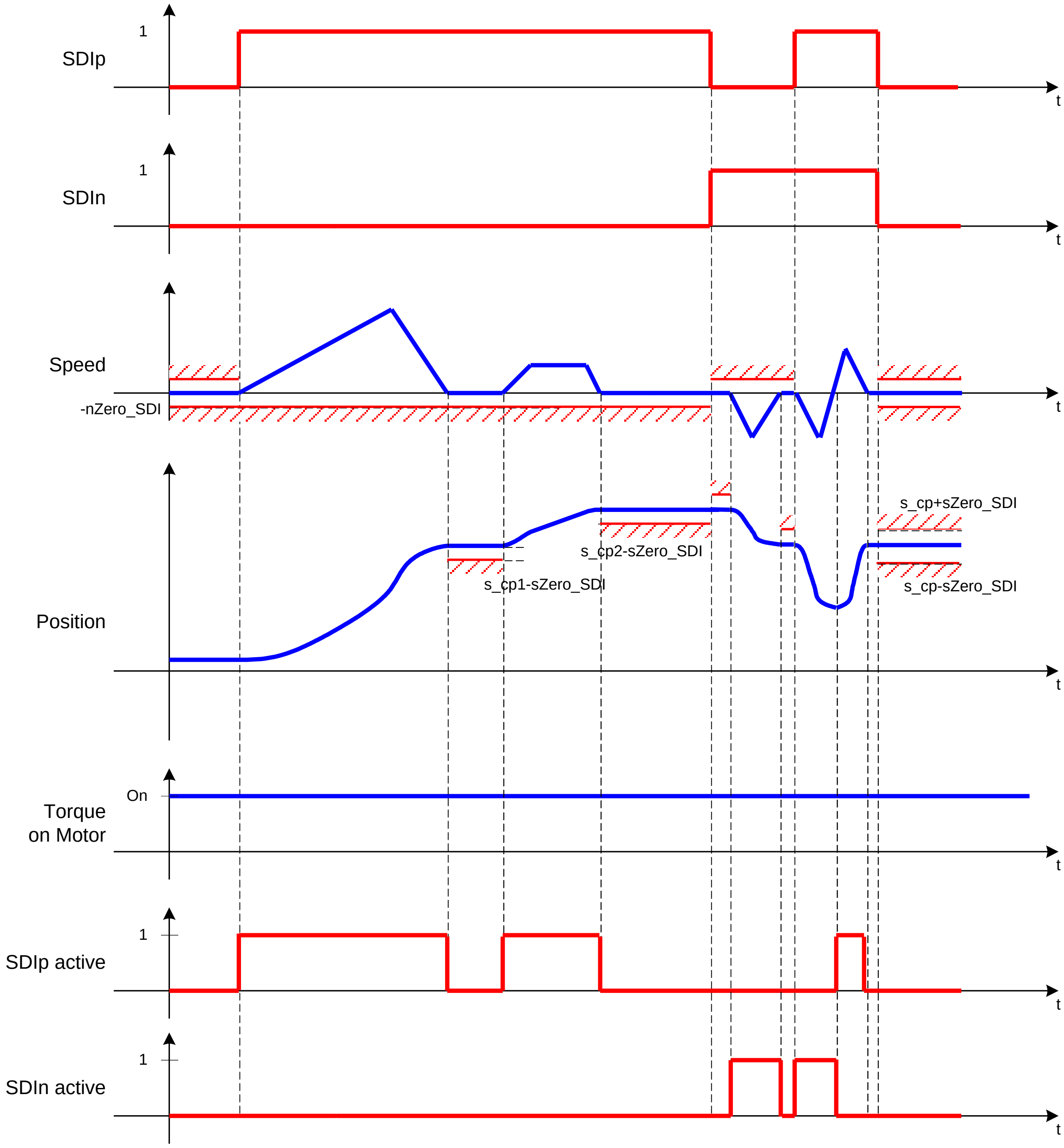

Figure 22: Functional principle of Safe Direction (SDI), example for positive direction

With setting the controlbit SDI the corresponding direction of the motor is enabled. The allowed direction is given by the Bits *SDI positive direction control* (SDIp) and *SDI negative direction control* (SDIn).

When reaching the speed zero the current position is latched (s_cp). The position shall be monitored not to exceed the *SDI position zero window* limit s_cp - s_Zero_SDI for positive direction and s_cp + s_Zero_SDI for negative direction.

For the speed zero the monitoring within the *SDI velocity zero window* limit 0 - n_Zero_SDI for positive direction and 0 + n_Zero_SDI for negative direction should optionally be monitored.

The SDI statusbits operates as a sensor: if the drive moves in positive direction Statusbit SDIp is active. If the drive moves in negative direction Statusbit SDIn is active.

8.14.2 Activation

Bit *SDI positive direction control* (SDIp) and *SDI negative direction control* (SDIn) in the safety controlword.

Table 35: State table for SDIp and SDIn

Controlword		Statusword		
SDIp	SDIn	SDIp	SDIn	Description
1	1	-	-	positive and negative direction enabled
1	0	1	0	only positive direction enabled
0	1	0	1	only negative direction enabled
0	0	0	0	no movement

8.14.3 Objects for SDI

The following objects are defined

Table 36: Objects for SDI

Object Code	Name	Type	M/O/C	Value
VAR	SDI positive direction command	BOOLEAN	O	Read: 0: drive does not moves in positive direction 1: drive moves in positive direction Write: 0: Disable positive direction 1: enable positive direction
VAR	SDI negative direction command	BOOLEAN	O	Read: 0: drive does not moves in negative direction 1: drive moves in negative direction Write: 0: Disable negative direction 1: enable negative direction
VAR	SDI position zero window u16	UNSIGNED16	C[1]	Position window for stop position Unit: [Position Units]
VAR	SDI position zero window u32	UNSIGNED32	C[1]	Position window for stop position Unit: [Position Units]
VAR	SDI velocity zero window u16	UNSIGNED16	O	Velocity window for n=0 Unit: [Velocity Units]
VAR	SDI velocity zero window u32	UNSIGNED32	O	Velocity window for n=0 Unit: [Velocity Units]

[1] At least one of these objects is mandatory, if the safety function is supported

8.14.4 Error reaction

Table 37: Error reaction SDI

Error reason	Behavior
SDI position zero window violated	STO
SDI velocity zero window violated	STO

8.15 Safe Speed Monitor SSM

8.15.1 Description

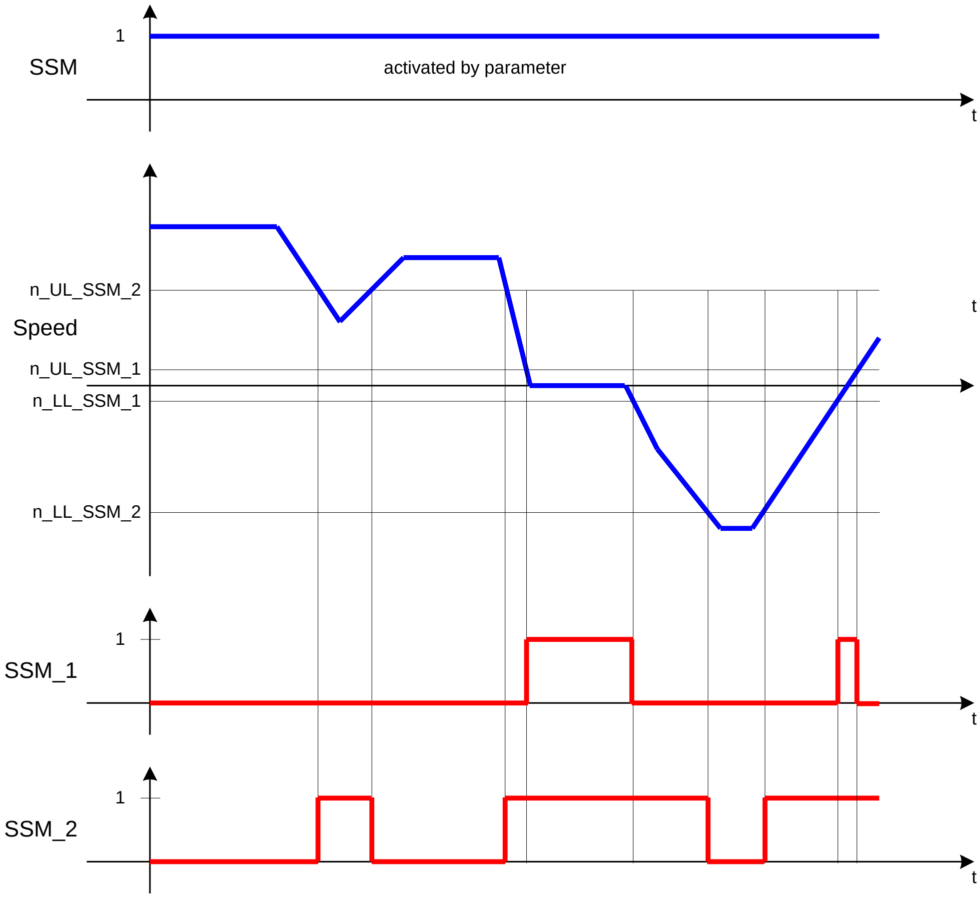

Figure 23: Functional principle of Safe Speed Monitor (SSM0, SSM1)

SSM can be implemented as local outputs or as bits in the safety statusword.

The output/bit SSM is set "1" if the speed is within the *SSM velocity upper limit* (n_UL_SSM) and *SSM velocity lower limit* (n_LL_SSM).

8.15.2 Activation

By parameterization:

If the objects n_UL_SSM and n_LL_SSM are supported and at least one is unequal zero SSM is activated.

8.15.3 Objects for SSM

The following objects are defined

Table 38: Objects for SSM

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SSM status	BOOLEAN	O	Read: 0: is not active 1: is active
ARRAY (see 3.1.1)	SSM velocity upper limit i16	INTEGER16	C[1]	Upper velocity limit Unit: [Velocity Units]

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SSM velocity upper limit i32	INTEGER32	C[1]	Upper velocity limit Unit: [Velocity Units]
ARRAY (see 3.1.1)	SSM velocity lower limit i16	INTEGER16	C[2]	Lower velocity limit Unit: [Velocity Units]
ARRAY (see 3.1.1)	SSM velocity lower limit i32	INTEGER32	C[2]	Lower velocity limit Unit: [Velocity Units]

[1] At least one of these objects is mandatory, if the safety function is supported

[2] At least one of these objects is mandatory, if the safety function is supported

The default usage for SSM_1 (the first instance of this function) shall be the monitoring of speed Zero.

8.15.4 Error Reaction

No error reaction

8.16 Safe CAM SCA

8.16.1 Description

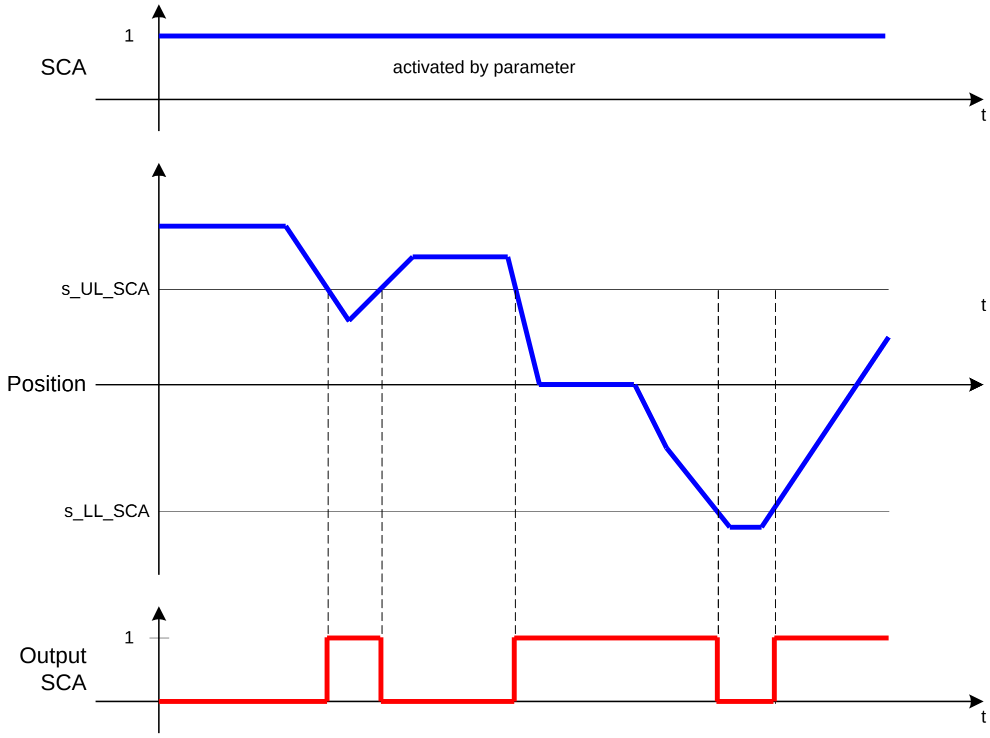

Figure 24: Functional principle of Safe CAM (SCA)

SCA can be implemented as a local output or as a bit in the safety statusword.

The output/bit SCA is set "1" if the position is within the *SCA position upper limit* (s_UL_SCA) and *SCA position lower limit* (s_LL_SCA).

8.16.2 Activation

By parameterization:

If the objects s_UL_SCA and s_LL_SCA are supported and at least one is unequal zero SCA is activated.

8.16.3 Objects for SCA

The following objects are defined

Table 39: Objects for SCA

Object Code	Name	Type	M/O/C	Value
ARRAY (see 3.1.1)	SCA status	BOOLEAN	O	Read: 0: is not active 1: is active
ARRAY (see 3.1.1)	SCA position upper limit i16	INTEGER16	C[1]	Upper position limit Unit: [Position Units]
ARRAY (see 3.1.1)	SCA position upper limit i32	INTEGER32	C[1]	Upper position limit Unit: [Position Units]
ARRAY (see 3.1.1)	SCA position lower limit i16	INTEGER16	C[2]	Lower position limit Unit: [Position Units]
ARRAY (see 3.1.1)	SCA position lower limit i32	INTEGER32	C[2]	Lower position limit Unit: [Position Units]

[1] At least one of these objects is mandatory, if the safety function is supported

[2] At least one of these objects is mandatory, if the safety function is supported

8.16.4 Error Reaction

No error reaction.