



Siemens  
Industry  
Online  
Support

APPLICATION EXAMPLE

# Safety programming in PLC Basic Program plus Application example

SINUMERIK ONE / CNC-SW 6.25 / Basic Program plus / Safety

**SIEMENS**

# Legal information

## Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. **The application examples are not subject to standard tests and quality inspections of a chargeable product and may contain functional and performance defects or other faults and security vulnerabilities. You are responsible for the proper and safe operation of the products in accordance with all applicable regulations, including checking and customizing the application example for your system, and ensuring that only trained personnel use it in a way that prevents property damage or injury to persons. You are solely responsible for any productive use.**

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. Any further use of the application examples is explicitly not permitted and further rights are not granted. You are not allowed to use application examples in any other way, including, without limitation, for any direct or indirect training or enhancements of AI models.

## Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

## Other information

Siemens reserves the right to make changes to the application examples at any time without notice and to terminate your use of the application examples at any time. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (<https://www.siemens.com/global/en/general/terms-of-use.html>) shall also apply.

## Cybersecurity information

Siemens provides products and solutions with industrial cybersecurity functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial cybersecurity concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial cybersecurity measures that may be implemented, please visit [www.siemens.com/cybersecurity-industry](http://www.siemens.com/cybersecurity-industry).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Cybersecurity RSS Feed under <https://www.siemens.com/cert>.

# Table of contents

<b>1.</b>	<b>Introduction .....</b>	<b>4</b>
1.1.	Overview .....	4
1.2.	Prerequisite knowledge .....	4
1.3.	Components used .....	4
1.4.	Scope of functions.....	5
<b>2.</b>	<b>Program structure .....</b>	<b>6</b>
2.1.	Structure.....	6
2.2.	SafetyInterface and mapping .....	6
2.2.1.	Functionality .....	6
2.2.2.	Superset capability .....	8
2.3.	Safety machine data model.....	8
2.4.	Program modules .....	9
2.4.1.	Separation into function modules .....	9
2.4.2.	Calling the function modules in controller blocks .....	10
2.5.	Data exchange between safety and standard programs .....	11
2.5.1.	Data exchange in the safety program .....	11
2.5.2.	Data exchange in the standard program .....	12
2.6.	Call structure.....	13
2.7.	Module functions .....	14
2.7.1.	Operating modes.....	14
2.7.2.	Emergency stop evaluation .....	14
2.7.3.	Door control.....	15
2.7.4.	Control of the safety-integrated functions of the drive .....	17
2.8.	Additional functions .....	20
2.8.1.	Test stop .....	20
2.8.2.	Brake test.....	21
2.8.3.	Messages .....	22
2.9.	Using the CMVM project .....	23
2.9.1.	Behavior Model .....	23
2.9.2.	Operation .....	24
<b>3.</b>	<b>Appendix .....</b>	<b>25</b>
3.1.	Service and support .....	25
3.2.	Links and literature .....	26
3.3.	Change documentation .....	26

# 1. Introduction

## 1.1. Overview

Machine safety has a central role in modern manufacturing. Protecting people and machines is an important priority and must be guaranteed in all operating conditions. Even with machine tools equipped with SINUMERIK ONE, it is essential to ensure reliable and safe shutdown in the event of a hazardous situation.

The programming of safety functions is a fundamental component of the safety concept and is carried out on the SIMATIC S7-1500F controller integrated into SINUMERIK ONE. When programming, it is therefore advisable to ensure that the safety program and the standard user program are structurally compatible so that they can interact with each other without any problems.

This application example shows a possible implementation of the structure and programming of the safety program for a SINUMERIK ONE on the SIMATIC S7-1500F controller.

As with the structure of the standard user program, the following points were considered when creating the application example:

- Reusability of the code
- Quick and easy expandability (scalability)
- Compliance with the programming style guide for SIMATIC S7 controllers
- Reduction of complexity
- Easy monitoring and troubleshooting



The program presented in this application example illustrates one possible way of implementing the control of safety functions. The target is to show how a program can be structured to keep it simple but also flexible and thus easily expandable.

When designing and programming the safety functions of a machine, the safety functions used, and their control must be specifically tailored/adapted to this machine in accordance with the applicable laws and standards.

## 1.2. Prerequisite knowledge

The knowledge listed in the documentation "109815160\_BasicProgramPlus\_AppNote\_ToolManagement\_doc\_v4\_en.pdf" also applies to this part of the application example. Additional knowledge is also required:

- **Basic knowledge of SIMATIC S7-Safety project planning and programming**  
You should already have basic knowledge of SIMATIC S7-1500F controller project planning and programming. Information on this topic can be found in the SIMATIC S7-Safety Project – Configuring and Programming manual ([13](#)).
- **Basic knowledge of SINUMERIK ONE Safety Integrated**  
This application example describes the control of the safety functions integrated in SINUMERIK ONE. It does not go into detail about how the Safety Integrated functions work. Information on SINUMERIK ONE Safety Integrated can be found in the manual ([14](#)).

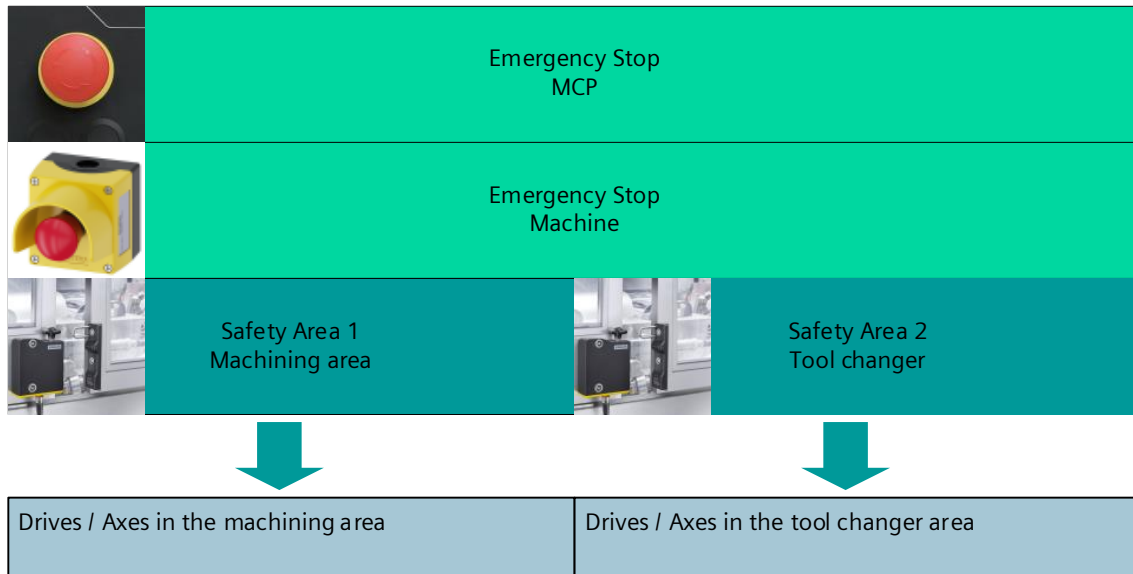
## 1.3. Components used

A list of the components used in the creation and components of the application example can be found in the main documentation "109815160\_BasicProgramPlus\_AppNote\_doc\_v4\_en.pdf".

## 1.4. Scope of functions

The safety program presented in this application example includes programming for a machine with the following safety-related components:

- 2 emergency stop switches (SS1 on all axes)
- 2 protective areas with safety doors (including locking/unlocking logic, SOS/SLS on axes in the respective area)
- Enable button for moving the axes at reduced speed (SLS) when the safety door is open
- Drives

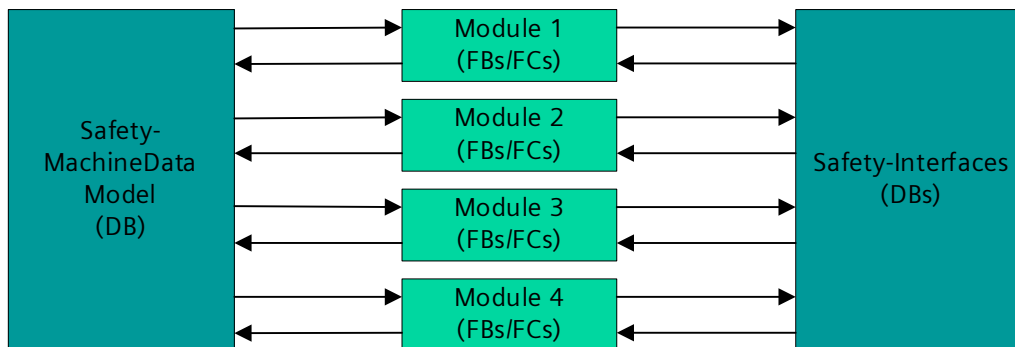


## 2. Program structure

### 2.1. Structure

Like the standard user program, the safety program is divided into different modules that represent individual functions/evaluations (e.g., EStop, control of Safety Integrated functions) and work independently of each other. Logical information is exchanged between the modules via a data interface—the SafetyMachineDataModel (DB).

Hardware signals, such as the safety telegrams of the drives and fail-safe peripheral signals of an ET200SP, are stored in separate interface data blocks and evaluated or described by the individual modules.

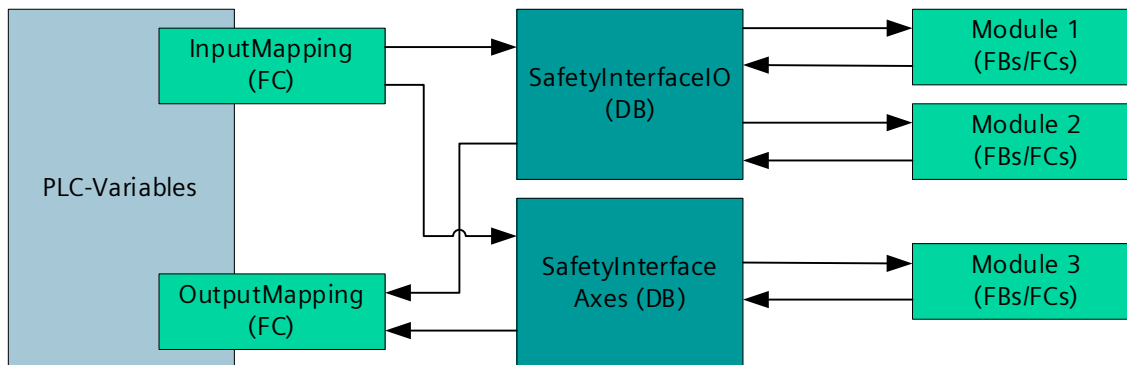


### 2.2. SafetyInterface and mapping

#### 2.2.1. Functionality

The project uses two safety interface DBs that contain variables for all signals that are available on the machine as hardware signals and are required for implementing the safety functions. The safety interfaces thus represent the PLC variables relevant to the safety program. In the application example, the following two safety interface datablocks are used, whose data is used by the program modules that require the data.

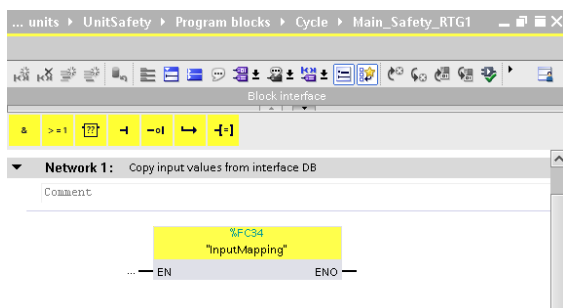
- **SafetyInterfaceIO**  
Contains variables that represent the PLC variables of IO signals, such as emergency stop buttons or variables for door control and evaluation.
- **SafetyInterfaceAxes**  
Contains variables that represent the drive telegrams.



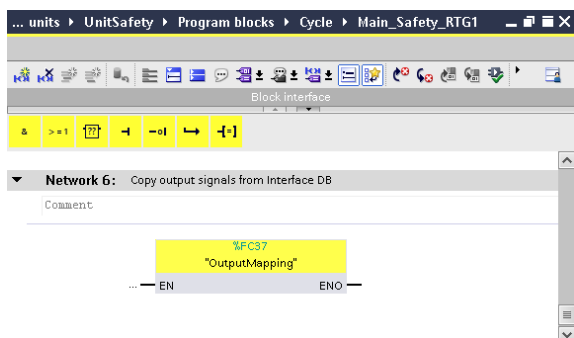
SafetyInterfaceAxes [Safety]		
	Name	Data type
1	Static	
2	SafetyTelAxisX1 Status	LDrvSafe_typeSina...
3	SafetyTelAxisX1 Control	LDrvSafe_typeSina...
4	SafetyTelAxisY1 Status	LDrvSafe_typeSina...
5	SafetyTelAxisY1 Control	LDrvSafe_typeSina...
6	SafetyTelAxisZ1 Status	LDrvSafe_typeSina...
7	SafetyTelAxisZ1 Control	LDrvSafe_typeSina...
8	SafetyTelAxisB1 Status	LDrvSafe_typeSina...
9	SafetyTelAxisB1 Control	LDrvSafe_typeSina...
10	SafetyTelAxisC1 Status	LDrvSafe_typeSina...
11	SafetyTelAxisC1 Control	LDrvSafe_typeSina...
12	SafetyTelAxisSP1 Status	LDrvSafe_typeSina...
13	SafetyTelAxisSP1 Cont...	LDrvSafe_typeSina...
14	SafetyTelAxisA1 Status	LDrvSafe_typeSina...
15	SafetyTelAxisA1 Control	LDrvSafe_typeSina...
16	SafetyTelAxisX2Status	LDrvSafe_typeSina...
17	SafetyTelAxisX2Control	LDrvSafe_typeSina...
18	SafetyTelAxisV1 Status	LDrvSafe_typeSina...
19	SafetyTelAxisV1 Control	LDrvSafe_typeSina...
20	SafetyTelAxisW1 Status	LDrvSafe_typeSina...
21	SafetyTelAxisW1 Control	LDrvSafe_typeSina...

SafetyInterfacelO [Safety]		
	Name	Data type
1	Static	
2	eStopMCP	Bool
3	eStopMachine	Bool
4	eStopReset	Bool
5	permissionSwitch	Bool
6	door1Closed	Bool
7	door2Closed	Bool
8	modeSetup	Bool
9	modeProduction	Bool
10	lockDoor1	Bool
11	lockDoor2	Bool
12	qbadSlot2	Bool

At the start of the safety cycle (Main\_Safety\_RTG1), the relevant PLC input variables are copied by the "InputMapping" block to the SafetyInterfaces so that they are then available for later use in the program.



At the end of a safety cycle, the values calculated by the safety program for the PLC output variables are copied from the SafetyInterfaces to the PLC output variables by the "OutputMapping" block.



### 2.2.2. Superset capability

This structure for mapping the hardware signals with the software is also used in the Simple Mapping Client and is described in more detail in the associated documentation ([18](#)). It enables different machine configurations to be mapped (superset capability).

For this purpose, all signals required for different machine configurations can be created in the SafetyInterfaces and can be used in the safety program. This is done regardless of whether they are available for the respective machine configuration (150% software).

In this concept, the hardware configuration is adapted to the machine (100% hardware) and only the available hardware signals are read or written for a specific machine configuration using the "InputMapping" and "OutputMapping" blocks. Signals in the Safety Interfaces that are required for other machine configurations are not written from the hardware or transferred to the hardware in this case.

It is possible to create copy templates for the "InputMapping" and "OutputMapping" blocks that contain all signals possible for the respective machine. These can then be copied into the respective project for a specific machine configuration. After the master copies have been inserted into a project, the program parts that are not required for this machine variant can be removed from the blocks in the project. These program parts are easy to identify, as compilation errors occur at these points due to the adapted hardware configuration.

**NOTE**

The Simple Mapping Client ([18](#)) is available for generating the "InputMapping" and "OutputMapping" blocks, which automatically creates these blocks via the Openness interface. This is not yet compatible with software units, which is why the alternative method with 150% master copies for "InputMapping" and "OutputMapping" is recommended for this application example.

### 2.3. Safety machine data model

The safety machine data model contains all logical data that is generated by the individual modules during program processing and is required by other modules for controlling the safety functions. The data is structured according to functions/machine components, so that the safety machine data model remains clear even in larger projects.

Furthermore, there are no direct dependencies between the individual data groups in the machine data model, so that changes, extensions, or even the addition of new "data groups" are possible without repercussions.

SafetyMachineDataModel [Safety]			
	Name	Data type	Start value
1	▼ Static		
2	▼ global	typeSafetyDataGlo...	
3	allAck	Bool	false
4	allAckReq	Bool	false
5	powerOnRdy	Bool	false
6	permissionSwitch	Bool	false
7	▶ modes	typeSafetyDataGlo...	
8	▶ eStop	typeSafetyDataGlo...	
9	▶ doors	typeSafetyDataGlo...	
10	▶ axes	typeSafetyDataGlo...	
11	▼ axes	typeSafetyDataAxes	
12	▶ X1	typeSafeAxis	
13	▶ Y1	typeSafeAxis	
14	▶ Z1	typeSafeAxis	
15	▶ X2	typeSafeAxis	
16	▶ SP1	typeSafeAxis	
17	▶ A1	typeSafeAxis	
18	▶ B1	typeSafeAxis	
19	▶ C1	typeSafeAxis	
20	▶ V1	typeSafeAxis	
21	▶ W1	typeSafeAxis	
22	▼ dataFromStandard	typeDataToSafety	
23	axisTestStopRequi...	Bool	false
24	requestDoor1	Bool	false
25	requestDoor2	Bool	false
26	▼ areas	typeSafetyDataAreas	
27	▶ area1	typeSafetyDataSin...	
28	▶ area2	typeSafetyDataSin...	

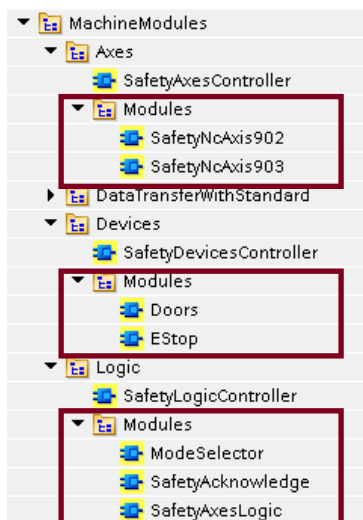
**NOTE**

Data for machine options that are not available in every machine configuration can also be created in the machine data model. If an option is not available on a machine, the data has no influence on the function of the machine (due to unlinked/uncreated hardware signals, see chapter [2.2](#)).

## 2.4. Program modules

### 2.4.1. Separation into function modules

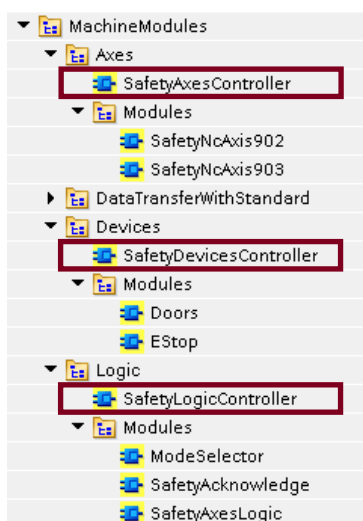
Individual safety functions for evaluation or control, such as emergency stops, doors, or control of the Safety Integrated functions of the drives, are divided into compact, independent function blocks that perform their respective functions without direct dependencies on each other. These blocks can therefore also be referred to as "function modules." These function modules can then be called multiple times as needed, for example, to use the function implemented in a function module for several axes.



### 2.4.2. Calling the function modules in controller blocks

The calls of the function modules are organized in so-called controller blocks, which divide the function modules into groups. The application example contains three controller blocks:

- SafetyDevicesController**  
 Contains the calls of the function modules for evaluating and controlling the input devices such as emergency stop buttons and safety doors. The calculated signals (e.g., AllEStopOK) are stored in the machine data model so that they can then be used by other blocks.
- SafetyLogicController**  
 Contains the calls of the function modules that contain the logic. The function modules contained here calculate the signals that are used later on for the direct control of the Safety Integrated functions of the axes.
- SafetyAxesController**  
 Contains the calls of the function modules for evaluating and controlling the PROFIsafe telegrams of the drives for all axes included in the project. Function modules are available in the project for drives that use telegram 902 and for drives that use telegram 903.



**NOTE** The function modules and controller blocks integrated in the application example show an example implementation and must be completed when used for a machine so that the control of the safety functions and the logic corresponds to the specific machine specifications.

**NOTE** Additional controller blocks and function modules (e.g., for compressed air, hydraulics, etc.) can be implemented in the project. The structure can be transferred from the blocks already implemented in the example.

## 2.5. Data exchange between safety and standard programs

Data exchange between the safety program and the standard program is implemented as described in the chapter "Data exchange with units" in the Safety Programming Guidelines (15). The following data blocks (DBs) are used in "UnitSafety" for this purpose.

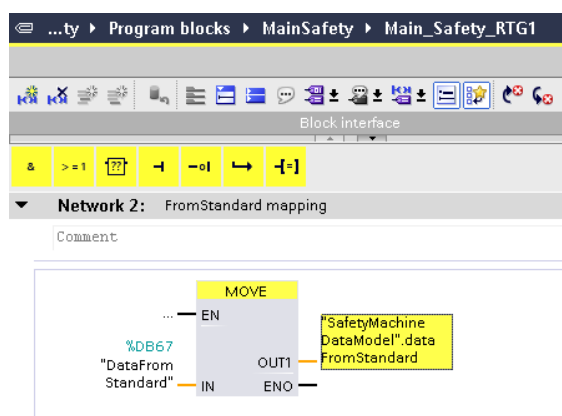
- DataFromStandard
- DataToSafety
- DataFromSafety

In addition to the structure specified by the programming guidelines, the data is incorporated into the safety machine data model and into the standard machine data model for better integration into the program structure. This integration is described in more detail in the following chapters.

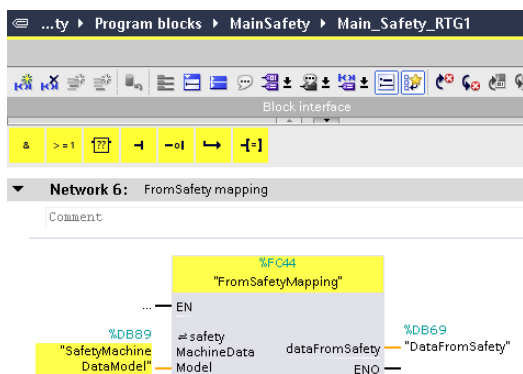
**NOTE** The data exchanged between the standard and safety programs can be expanded as needed. To do this, the required data must be expanded to the data types "typeDataFromSafety" or "typeDataToSafety." The data can then be used in the program.

### 2.5.1. Data exchange in the safety program

To make it easier to use the data in the program, the "FromStandard" data is also copied to the safety machine data model at the beginning of the safety program cycle.

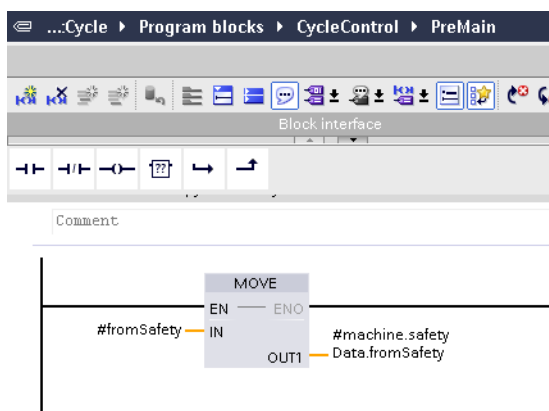


At the end of the safety program cycle, the required data is copied from the safety machine data model to the "DataFromSafety" data block. The FC "FromSafetyMapping" is used for this purpose.



### 2.5.2. Data exchange in the standard program

As in the safety program, the data exchanged between the program parts is integrated into the standard machine data model. Before the standard program is processed, the data coming from the safety program is copied into the machine data model in "PreMain".



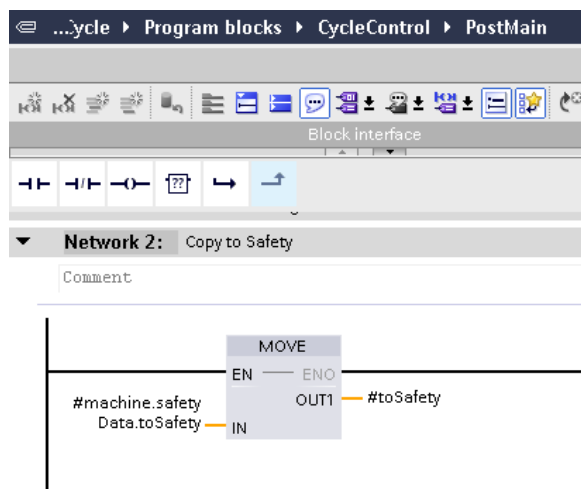
The data in the machine data model can then be used during program processing (1), and data is transferred to the safety program can be written to the machine data model (2).

```

...vare units > UnitMachineFunctions [MachineFunctions] > Program blocks > MachineModules > ModMcp > ModMcpFunctions > Functions > NcModMcpOemKeyExample
Block Interface
IF... CASE... FOR... WHILE... REGION
OF... TO DO... DO... (*...*)
81 REGION doors
82 REGION Door1
83 //Trigger Button pressed
84 #instTrigRequestDoor1(CLK := #machine.modMcp.fromMcp.KeyPad0em2.key3);
85
86 IF NOT #machine.safetyData.toSafety.requestDoor1 AND #instTrigRequestDoor1.Q THEN
87 //Request_door
88 #machine.safetyData.toSafety.requestDoor1 := TRUE;
89 ELSE #machine.safetyData.toSafety.requestDoor1 := #instTrigRequestDoor1.Q OR NOT #machine.safetyData.fromSafety.globalData.doors.door1.locked THEN
90 //Reset_door_request
91 #machine.safetyData.toSafety.requestDoor1 := FALSE;
92 END_IF;
    
```

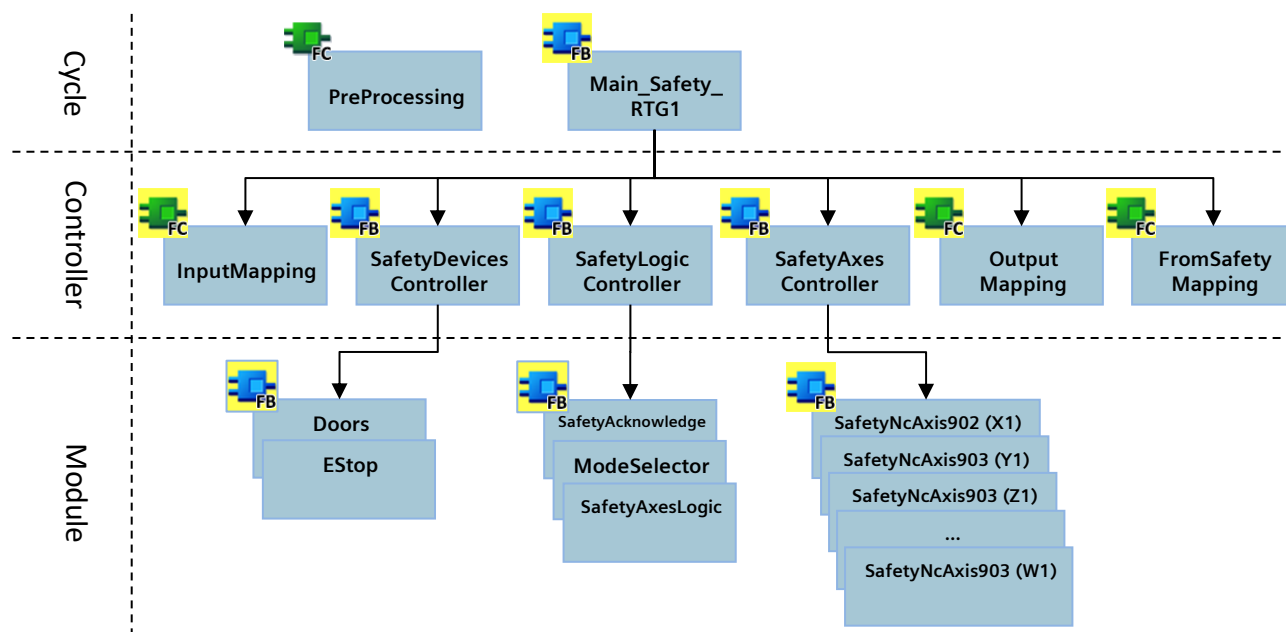
The code snippet shows a ladder logic program. Line 88 is highlighted with a red box and labeled '2', representing the assignment of data from the machine data model to the safety program. Line 89 is also highlighted with a red box and labeled '2', representing the assignment of data from the safety program to the machine data model. Line 91 is highlighted with a red box and labeled '1', representing the assignment of data from the machine data model to the safety program.

At the end of a program cycle in "PostMain," the "toSafety" data written in the standard machine data model is copied from the machine data model to the corresponding "toSafety" data block and thus transferred to the safety program, where it can then be used.



## 2.6. Call structure

The following graphic shows the call structure of the blocks used in the safety program.



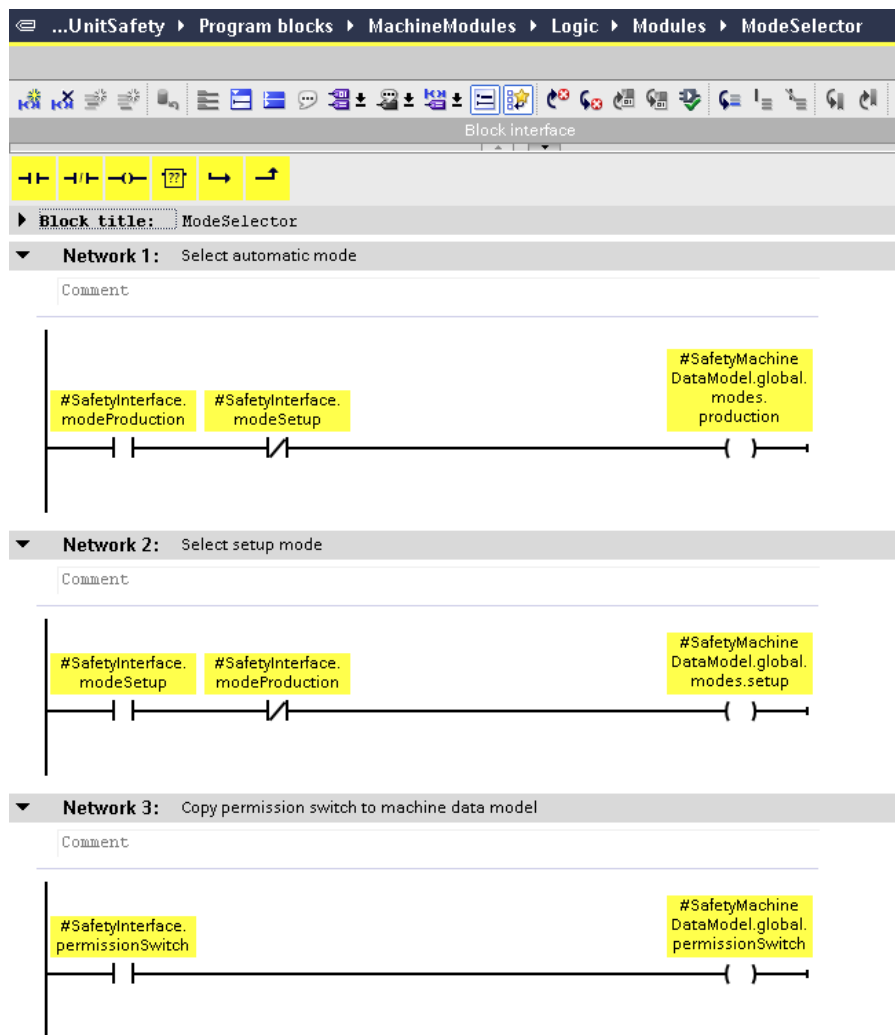
**NOTE** The clear structure of the individual controllers and modules makes it easy to add further modules or controllers or to adapt the existing blocks later.

**NOTE** The data required for editing the individual blocks, such as the safety machine data model and the safety interface, is transferred to the individual blocks via the InOut interface in accordance with the styleguide when the blocks are called.

## 2.7. Module functions

### 2.7.1. Operating modes

In the example project, two operating modes ("Setup" and "Production") are implemented in the safety program. These can be switched via digital inputs (e.g., via a key switch). The inputs are evaluated by the "ModeSelector" block, which is called in the "SafetyLogicController," and entered in the machine data model so that they can be used later by other blocks.

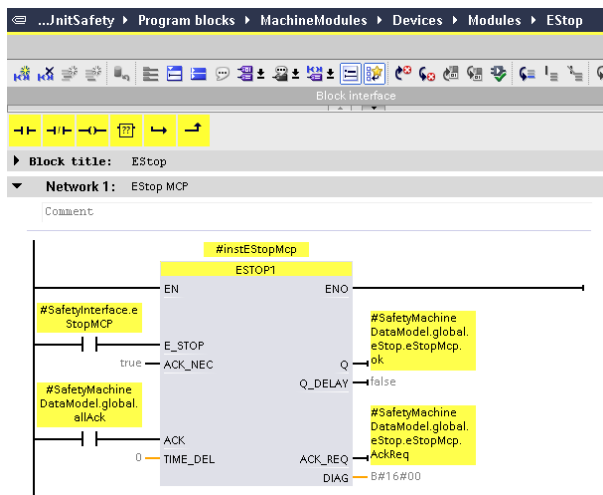


### 2.7.2. Emergency stop evaluation

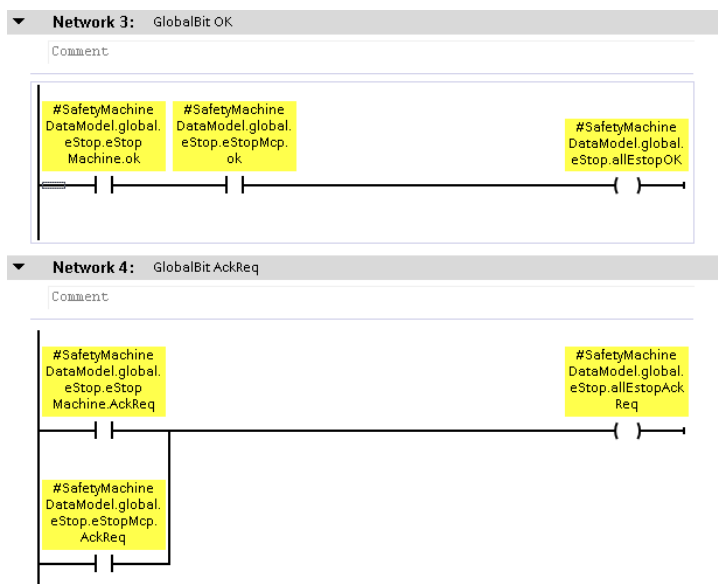
The "ESTOP1" system block is used to evaluate the emergency stop switches (digital inputs).

**NOTE** Information on the "ESTOP1" block can be found, for example, in the TIA Portal online help.

This block sets the "ok" and "AckReq" bits of the respective emergency stop in the machine data model, which can then be used later during further program processing.



In addition, the global bits for "allEstopOk" and "allEstopAckReq" are set in the "EStop" block in the machine data model to enable simple global evaluation of all emergency stops in the rest of the program.



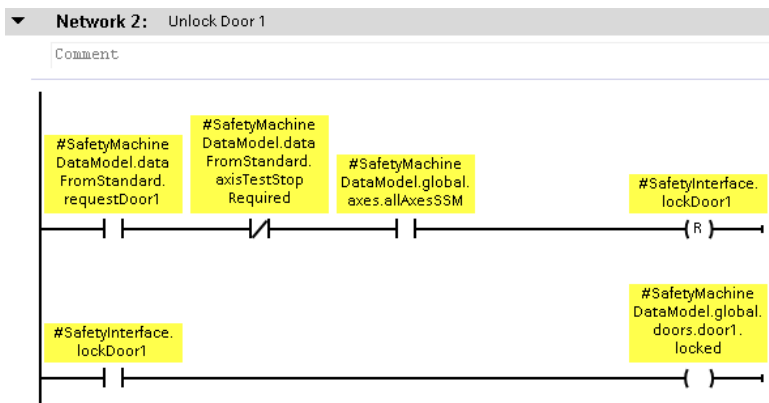
### 2.7.3. Door control

The example project implements a door control for safety doors. This may need to be adapted or expanded for use on a machine in order to meet specific safety requirements.

The evaluation and control of the two safety doors integrated into the project is carried out in the "Doors" block, which is called in the "SafetyDevicesController" block. The evaluation is carried out using the "SFDOOR" system block.

**NOTE** Information on the "SFDOOR" block can be found, for example, in the TIA Portal online help.

The safety doors in the example project can be locked or unlocked via digital output. The doors can be unlocked by pressing a button on the MCP (evaluation in the standard program and transfer of the request via "DataFromStandard"). In order for the doors to be unlocked, no test stop must be required on any of the axes integrated in the project and all axes must be at a standstill.



**NOTE** Like the "EStop" block, global bits for "allDoorsOK", "allDoorsLocked", and "doorsAckReq" are also set in the "Doors" block.

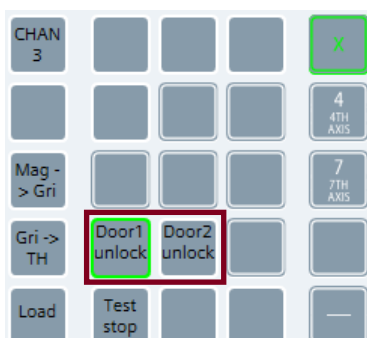
The logic for requesting door unlocking via the MCP and the behavior of the LED on the MCP is implemented in the standard program in the "NcModMcpOemKeyExample" block.

```

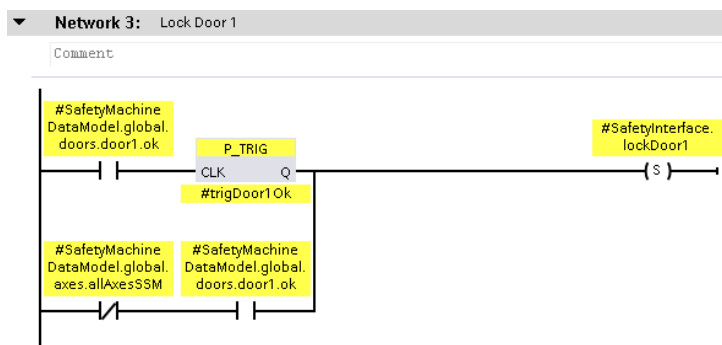
... Software units > UnitMachineFunctions [MachineFunctions] > Program blocks > MachineModules > ModMcp > ModMcpFunctions > Functions > NcModMcpOemKeyExample
Block interface
IF... CASE... FOR... WHILE... (...*) REGION
82 REGION Door1
83 //Trigger Button pressed
84 #instTrigRequestDoor1(CLK := #machine.modMcp.fromMcp.KeyPadOem2.key3);
85
86 IF NOT #machine.safetyData.toSafety.requestDoor1 AND #instTrigRequestDoor1.Q THEN
87 //Request door
88 #machine.safetyData.toSafety.requestDoor1 := TRUE;
89 ELIF #machine.safetyData.toSafety.requestDoor1 AND #instTrigRequestDoor1.Q OR NOT #machine.safetyData.fromSafety.globalData.doors.door1.locked THEN
90 //Reset door request
91 #machine.safetyData.toSafety.requestDoor1 := FALSE;
92 END_IF;
93
94 #machine.modMcp.toMcp.KeyPadOem2.led3 := (#machine.safetyData.toSafety.requestDoor1 AND #machine.frequency.clockFast) OR NOT #machine.safetyData.fromSafe
95 END_REGION
96

```

The request to unlock the door can also be set if the conditions are not met. In this case, the button on the MCP flashes and the door is only unlocked once the conditions are met. The request can also be canceled beforehand by pressing the button on the MCP again. Once the door has been unlocked, the button on the MCP lights up continuously.



The door is automatically locked again after it has been closed and acknowledged without error. If an axis is moved after the door has been unlocked but not yet opened, the door is automatically locked again and the unlock request must be set again.



### 2.7.4. Control of the safety-integrated functions of the drive

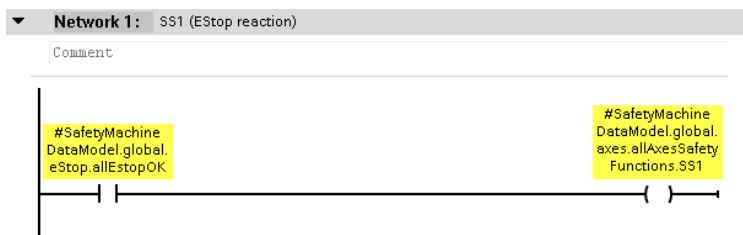
Several blocks interact with each other to control the Safety Integrated functions of the drives in the project. The blocks for evaluating the safety doors and emergency stops enter the corresponding signals into the safety machine data model so that they can be used in the "SafetyAxesLogic" block. The "SafetyAxesLogic" block uses this information to evaluate which Safety Integrated functions must be triggered on which drive.

**NOTE**

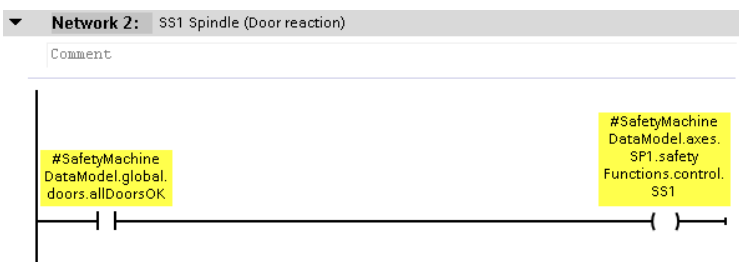
The logic in the example project is structured in such a way that the various control options are shown. The safety functions used and their control are usually machine-dependent. Therefore, the "SafetyAxesLogic" block must be adapted so that the control corresponds to the respective machine specifications.

Various variables are integrated into the machine data model to trigger the Safety Integrated functions of the drives:

- Triggering Safety Integrated functions on all drives/axes**  
 The respective Safety Integrated function can be triggered on all drives in the project using the variables "SafetyMachineDataModel.global.axes.allAxesSafetyFunctions.<Safety Integrated function>". The functions behave as FALSE-active.  
 In the example project, as soon as one of the emergency stop switches is pressed (SafetyMachineDataModel.global.eStop.allEStopOK = FALSE), the signal for SS1 is also set to "FALSE" on all axes, thereby triggering SS1 on all drives/axes.



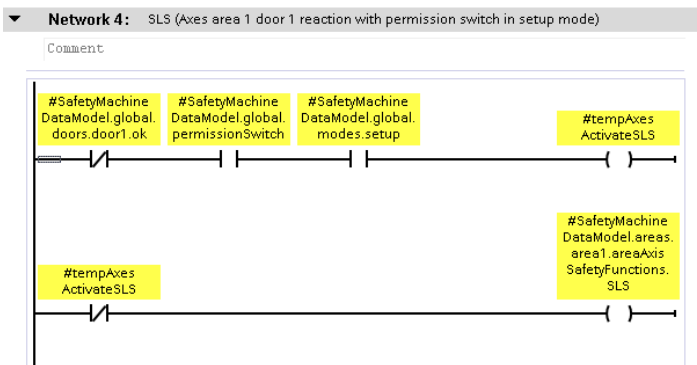
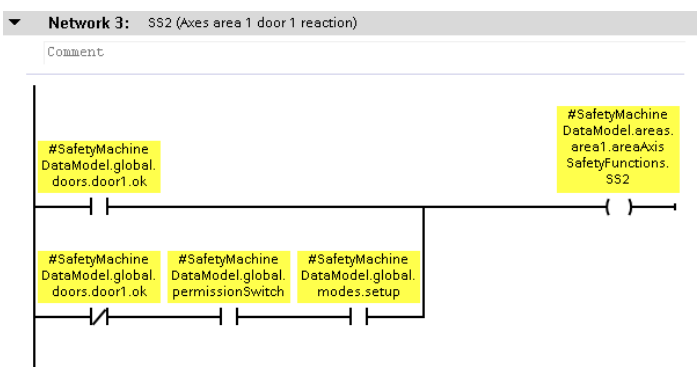
- Triggering Safety Integrated functions on a specific drive/axis**  
 The variable "SafetyMachineDataModel.axes.<axis>.safetyFunctions.control.<Safety Integrated function>" can be used to trigger the respective Safety Integrated function only for the respective drive/axis.  
 In the example project, as soon as one of the doors is opened (SafetyMachineDataModel.global.doors.allDoorsOK = FALSE), the signal for SS1 of the spindle is also set to "FALSE," thereby triggering SS1 on the spindle only.



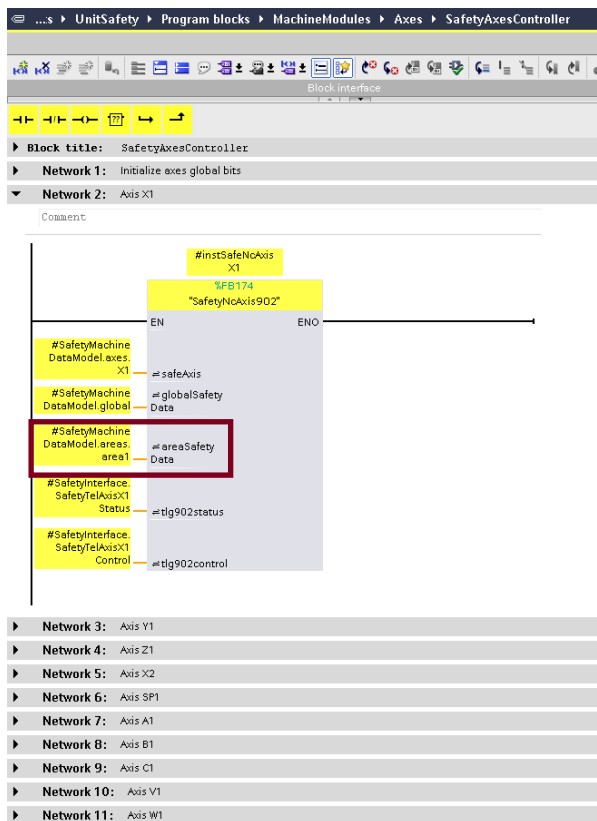
• **Triggering Safety Integrated functions on a group of axes (in an area)**

Using the variable "SafetyMachineDataModel.areas.<area>.areaAxisSafetyFunctions.<Safety Integrated function>", the respective Safety Integrated function can only be triggered for the drives of the respective area. The drives/axes are assigned to an area by calling the control blocks for the axes (see following section).

In the example project, when the door of an area is opened (SafetyMachineDataModel.global.doors.door<X>.), SS2 is triggered on the associated drives/axes. If the respective door is open and the machine is in "Setup" mode and the enable button is pressed, SLS is activated on the axes of the respective area instead. This allows the axes to be moved at a safe speed even when the door is open.

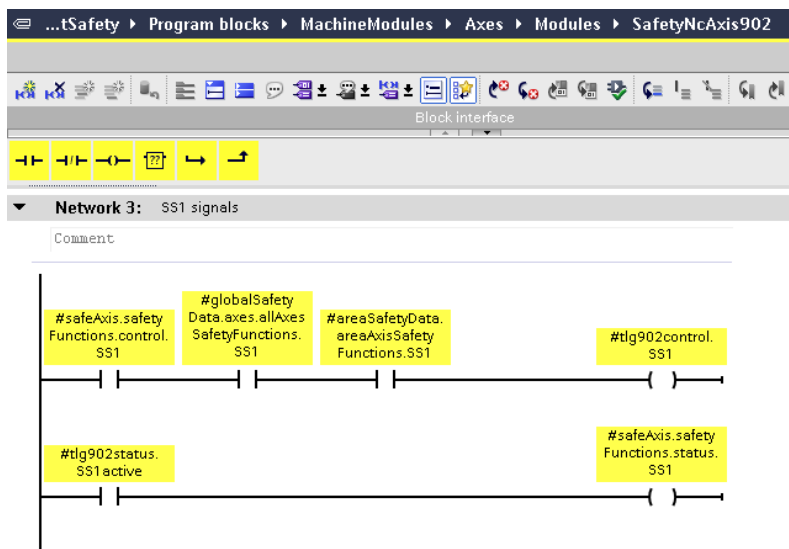


The values set in the machine data model are then evaluated by the "SafetyNcAxis902" (for drive telegram 902) or "SafetyNcAxis903" (for drive telegram 903) blocks. These are called once in the "SafetyAxesController" block for each drive/axis in the project.



When the block is called, the corresponding data for the drive/axis is transferred from the safety machine data model and SafetyInterface to the block. The area in which the respective drive/axis is located is also specified by connecting the respective area (area<X>) to the input parameter "areaSafetyData".

Within the blocks, the signals previously set in the "SafetyAxesLogic" block are evaluated and the result is transferred to the drive telegram.

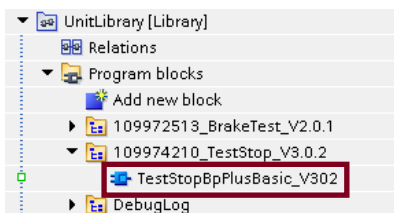


**NOTE** As soon as one of the signals (single-axis control, group control, or global control) of an axis is reset for one of the Safety Integrated functions, the respective Safety Integrated function is triggered at the drive. It is therefore also possible to use multiple triggers for different states.

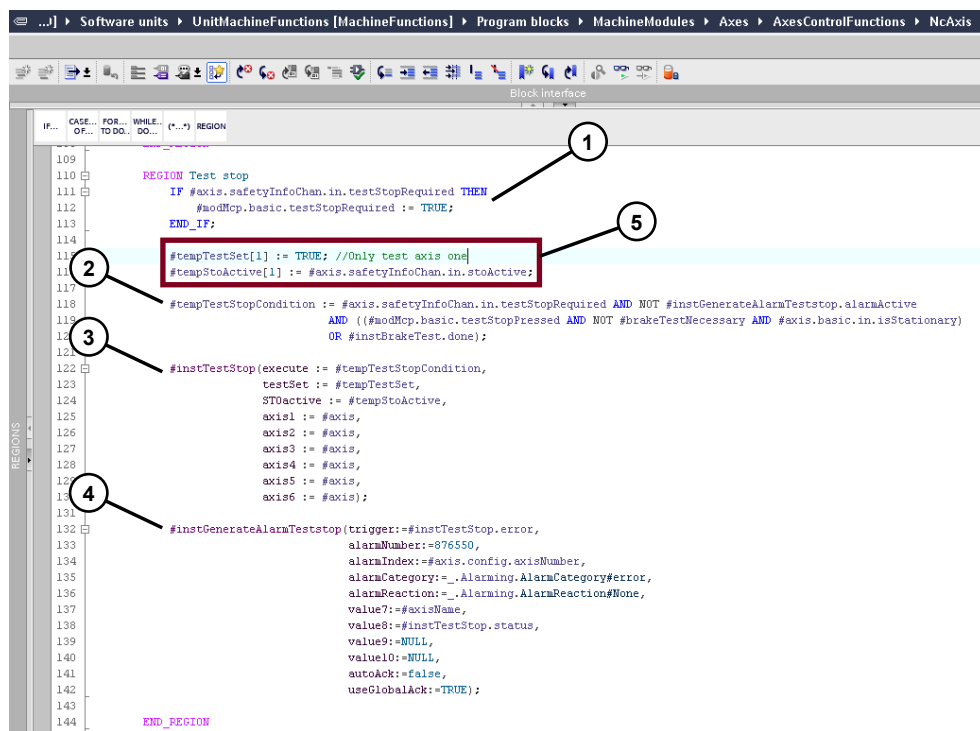
## 2.8. Additional functions

### 2.8.1. Test stop

To ensure that the shutdown paths of the drive devices are working correctly, a test stop must be performed regularly. An application example is already available for this in the SiePortal ([6](#)), whose modules are integrated into the existing software structure in this application example.



The block for the test stop is a standard block (not fail-safe), as the test stop is executed via the SafetyInfoChannel. For this reason, the block call is also integrated into the "NcAxis" block in the standard program.



1. For the MCP, a bit for "TestStopRequired" is set in the machine data model as soon as a test stop is required on an axis. This bit can be used, for example, to make the LED of the corresponding button on the MCP flash.
2. The conditions for starting the test stop are created using a temporary variable for greater clarity. The conditions for starting the test stop can be expanded if necessary.
3. Calling the test stop block
4. If an error occurs while editing the test stop block, an alarm is generated. This alarm contains additional values such as the name of the respective axis and the status of the test stop block, which can be used for diagnostics.

**NOTE**

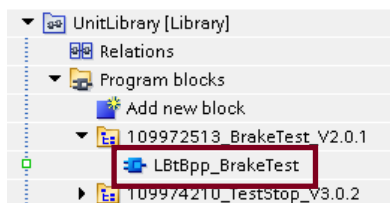
The test stop block is designed for up to 6 axes. However, since it is only ever executed for one axis due to the software structure in the application example, only the first element (5) is always written to the input arrays for the test set and the STO status of the axes.

The test stop can be executed via the "Test Stop" button (OEM keypad) on the MCP. Pressing the button sets the variable "MachineDataModel.modMcp.basic.testStopPressed," which is part of the conditions for the test stop.

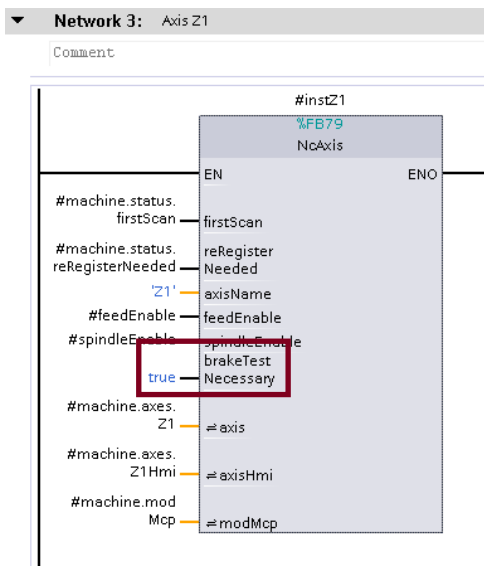


**2.8.2. Brake test**

For drives that have a brake, it may be necessary to check the brake regularly for proper functioning and to perform a so-called "brake test". Just like the test stop, the brake test is performed via the SafetyInfoChannel. There is also another application example for the brake test ([L7](#)) in the SiePortal, whose block is used in this application example.



Just like the test stop block, the call of the brake test block is integrated into the "NcAxis" block in the standard program. Since the brake test does not usually have to/can be performed on all axes, it must be activated via the input parameter "brakeTestNecessary" = TRUE on the "NcAxis" block.



If the brake test is activated on the respective axis, the test stop is not executed immediately when the conditions are met. In this case, the brake test is first performed on the axis. The test stop is then executed automatically as soon as the brake test has been completed. Just as with the test stop, if an error occurs while processing the brake test block, a message is issued that contains additional information such as the respective axis name and status of the brake test block.

```

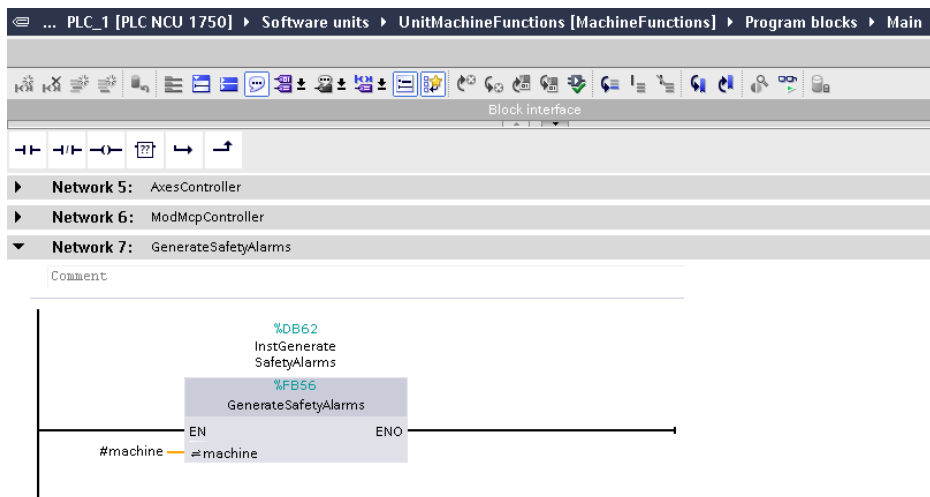
...eFunctions [MachineFunctions] > Program blocks > MachineModules > Axes > AxesControlFunctions > NcAxis
Block Interface
IF... CASE... FOR... WHILE... (*...) REGION
OF... TO DO... DO...

87
88 REGION Brake test
89 IF #brakeTestNecessary THEN
90 #tempBrakeTestCondition := #axis.basic.in.isStationary AND NOT #axis.basic.in.disabled
91 AND NOT #instGenerateAlarmBrakeTest.alarmActive;
92
93 #instBrakeTest(execute := #modMcp.basic.testStopPressed AND #tempBrakeTestCondition,
94 activateAutoDir := true,
95 axis := #axis);
96
97 #instGenerateAlarmBrakeTest(trigger := #instBrakeTest.error,
98 alarmNumber := 876551,
99 alarmIndex := #axis.config.axisNumber,
100 alarmCategory := _Alarming.AlarmCategory#error,
101 alarmReaction := _Alarming.AlarmReaction#None,
102 value7 := #axisName,
103 value8 := #instBrakeTest.status,
104 value9 := NULL,
105 value10 := NULL,
106 autoAck := false,
107 useGlobalAck := TRUE);
108 END_IF;
109 END_REGION
110

```

### 2.8.3. Messages

To inform the operator about the status of the safety devices, the "GenerateSafetyAlarms" block is available in "UnitMachineFunctions."



This evaluates the "FromSafety" values stored in the machine data model and generates alarms based on various states, for example to indicate which of the safety devices has been triggered.

Alarms			
Date	Delete	Number	Text
07/25/25 7:59:00.052		876560	EStop MCP triggered. Check button and acknowledge safety
07/25/25 7:58:59.152		876570	Door 1 open. Close door and acknowledge safety or switch to setup mode and use permission switch

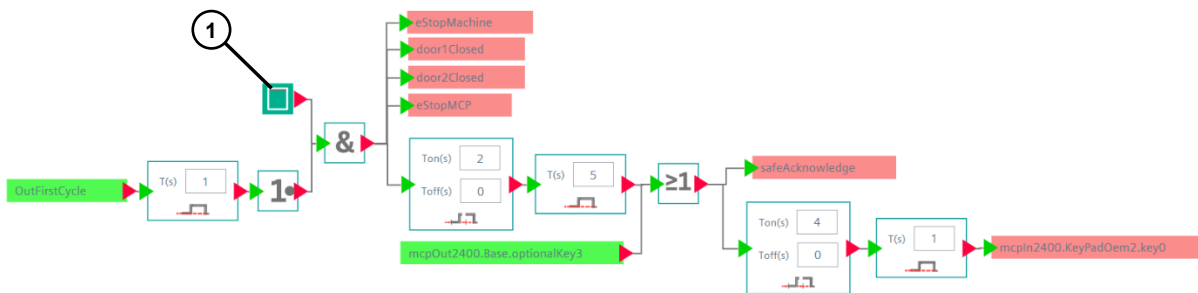
**NOTE** The "GenerateSafetyAlarms" block only contains examples of alarms when the safety devices in the project are triggered. It can be expanded with additional alarms if necessary.

## 2.9. Using the CMVM project

### 2.9.1. Behavior Model

Usually, the signals from the safety devices must be set and acknowledged after the controller has started up. In addition, a test stop must be performed after the controller has started up. For these tasks, a behavior model has been implemented in the Create myVirtual Machine project that automatically performs these tasks after startup, so that no alarms are pending after the controller has started up and the project can be used immediately.

The switch (1) allows you to deactivate the behavior model if you do not need it or want to test the startup process manually.



The behavior model performs the following sequence, among other things:

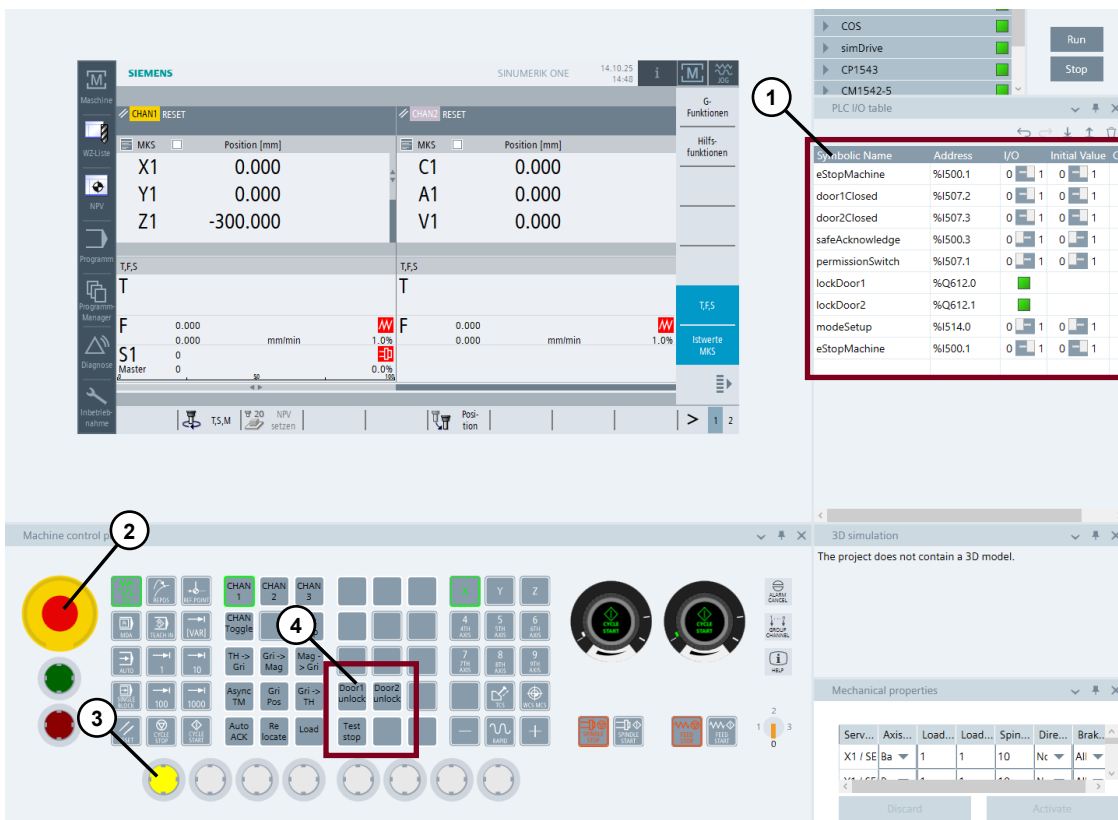
1. Setting the safety-relevant peripheral signals (eStopMachine, door1Closed, door2Closed, eStopMCP)
2. After a two-second delay: Setting "safeAcknowledge" for five seconds
3. After a further four-second delay: Set "KeyPadOem2.key0" (TestStop) for one second to perform a test stop

### 2.9.2. Operation

To operate the safety-related peripheral signals, you can use the PLC I/O table in Creaty myVirtualMachine (1). Here you can reset the signals to trigger the respective safety function. You can also trigger the eStopMcp via the MCP's emergency stop button (2). After triggering a safety function, the respective digital input must be reset and acknowledged via "safeAcknowledge".

You can also use the yellow button on the MCP (3) to acknowledge the safety program (safeAcknowledge). This also affects the behavior model, which sets the peripheral signal "safeAcknowledge" when the button is pressed.

In addition, you have the option of performing a test stop using the buttons on the second OEM keypad and setting the requests to open the doors (4).



## 3. Appendix

### 3.1. Service and support

#### SiePortal

The integrated platform for product selection, purchasing and support - and connection of Industry Mall and Online support. The SiePortal home page replaces the previous home pages of the Industry Mall and the Online Support Portal (SIOS) and combines them.

- Products & Services  
In Products & Services, you can find all our offerings as previously available in Mall Catalog.
- Support  
In Support, you can find all information helpful for resolving technical issues with our products.
- mySieportal  
mySiePortal collects all your personal data and processes, from your account to current orders, service requests and more. You can only see the full range of functions here after you have logged in.

You can access SiePortal via this address: [sieportal.siemens.com](https://sieportal.siemens.com)

#### Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers – ranging from basic support to individual support contracts.

Please send queries to Technical Support via Web form: [support.industry.siemens.com/cs/my/src](https://support.industry.siemens.com/cs/my/src)

#### SITRAIN – Digital Industry Academy

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that's tailored to the customer's specific needs.

For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page: [siemens.com/sitrain](https://siemens.com/sitrain)

#### Industry Online Support app

You will receive optimum support wherever you are with the "Industry Online Support" app. The app is available for iOS and Android:



## 3.2. Links and literature

No.	Topic
111	Siemens Industry Online Support <a href="https://support.industry.siemens.com">https://support.industry.siemens.com</a>
121	Link to the application example page <a href="https://support.industry.siemens.com/cs/ww/en/view/109815160">https://support.industry.siemens.com/cs/ww/en/view/109815160</a>
131	SIMATIC Industrial Software SIMATIC Safety – Configuring and Programming <a href="https://support.industry.siemens.com/cs/ww/en/view/54110126">https://support.industry.siemens.com/cs/ww/en/view/54110126</a>
141	SINUMERIK ONE Safety Integrated <a href="https://support.industry.siemens.com/cs/ww/en/view/109994175">https://support.industry.siemens.com/cs/ww/en/view/109994175</a>
151	Programming Guidelines and Programming Styleguide for SIMATIC S7-1200 and S7-1500, and WinCC (TIA Portal) <a href="https://support.industry.siemens.com/cs/ww/en/view/81318674">https://support.industry.siemens.com/cs/ww/en/view/81318674</a>
161	SINUMERIK ONE: Testing shutdown paths <a href="https://support.industry.siemens.com/cs/ww/en/view/109974210">https://support.industry.siemens.com/cs/ww/en/view/109974210</a>
171	SINAMICS brake test with SINUMERIK ONE – Safety Integrated <a href="https://support.industry.siemens.com/cs/ww/en/view/109801664">https://support.industry.siemens.com/cs/ww/en/view/109801664</a>
181	SINUMERIK ONE Create MyWorkflow – Engineering & Service Workflow (Simple Mapping) <a href="https://support.industry.siemens.com/cs/ww/en/view/109925843/171404256139?dl=de">https://support.industry.siemens.com/cs/ww/en/view/109925843/171404256139?dl=de</a>

## 3.3. Change documentation

Version	Date	Modification
V4.0	10/2025	First edition, with V4 of the application example for PLC Basic Program plus