

**SIEMENS**

*Ingenuity for life*

*Industry Online Support*

Home

# Kinematics Language for Machining Centers Simulation with SIMIT, PLCSIM Advanced & NX MCD

Product / version / specification / keyword

<https://support.industry.siemens.com/cs/ww/en/view/109804122>

Siemens  
Industry  
Online  
Support



## 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. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.

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. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

### 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. 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://support.industry.siemens.com>) shall also apply.

### Security information

Siemens provides products and solutions with Industrial Security 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 security 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 security measures that may be implemented, please visit <https://www.siemens.com/industrialsecurity>.

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 Security RSS Feed at: <https://www.siemens.com/industrialsecurity>.

# Table of contents

	<b>Legal information</b> .....	<b>2</b>
<b>1</b>	<b>Introduction</b> .....	<b>4</b>
1.1	Overview.....	4
1.2	Mode of operation .....	6
1.3	Components used .....	6
<b>2</b>	<b>Software setup</b> .....	<b>7</b>
2.1	TIA Portal .....	7
2.2	SIMIT .....	8
2.3	NX Mechatronics Concept Designer .....	11
2.3.1	Overview.....	12
<b>3</b>	<b>Engineering</b> .....	<b>13</b>
3.1	Description of interface.....	13
3.1.1	APP_LKinLang_SpindleControl .....	13
3.1.2	APP_LKinLang_auxiliaryAxis .....	15
3.1.3	Relevance of FlagModes.....	17
3.2	Project integration .....	18
3.2.1	Requirements .....	18
3.2.2	Import of application .....	19
3.3	Operation.....	20
3.3.1	APP_LKinLang_SpindleControl .....	20
3.3.2	APP_LKinLang_auxiliaryAxis .....	21
3.4	Error handling.....	22
3.4.1	APP_LKinLang_SpindleControl – Error identifiers.....	22
3.4.2	APP_LKinLang_SpindleControl – Error reaction and acknowledgement .....	22
3.4.3	APP_LKinLang_auxAxis .....	22
<b>4</b>	<b>Operating the Digital Twin</b> .....	<b>23</b>
4.1	Starting the simulation.....	23
4.1.1	NX MCD .....	23
4.1.2	SIMIT .....	24
4.1.3	WinCC Runtime.....	25
4.2	Operating the simulation .....	26
4.2.1	Program Management – File Transfer .....	27
4.2.2	Program Management - Selecting a program .....	28
4.2.3	Program Management – Editing a program .....	29
4.2.4	Program execution .....	30
4.2.5	Simulation of safety functionality .....	32
4.2.6	Manual control of kinematic.....	33
4.2.7	Toggle Sawblade ↔ Waterjet display .....	34
4.3	Error handling.....	35
4.3.1	Resetting of errors (startup & safety) .....	35
<b>5</b>	<b>Appendix</b> .....	<b>36</b>
5.1	Service and support .....	36
5.2	Application support.....	37
5.3	Links and literature .....	37
5.4	Change documentation .....	37

# 1 Introduction

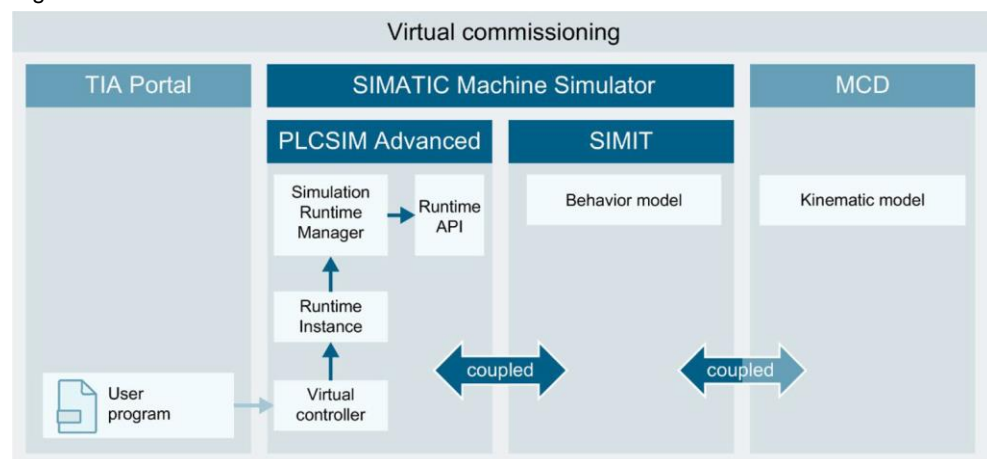
This application example allows simulating a digital twin of a bridge saw with SIMATIC Kinematics Language. The functionality of the textual program interpretation is supplemented by additional support for machining specific requirements – especially the usage of spindle-based operations.

The application uses SIMATIC PLCSIM Advanced, SIMIT and NX Mechatronics Concept Designer (MCD). The general usage structure of these virtual commissioning tools is depicted in [Figure 1-1](#).

The MCD model consists of a Cartesian portal kinematic with a rotary orientation axis and a spindle. Furthermore, a SIMIT model is providing a simulation of the machine behavior. SIMATIC PLCSIM Advanced is used to emulate a SIMATIC S7-1500T controller.

The combination of these software components enables a Software in the Loop (SiL) based virtual commissioning of the digital twin.

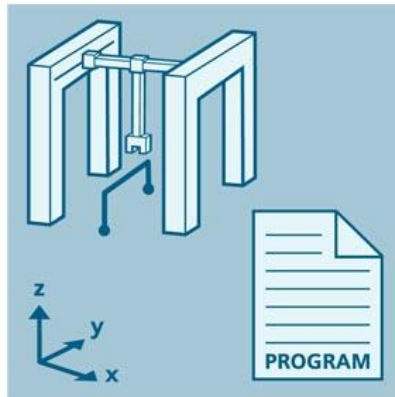
Figure 1-1 Schematic SiL architecture



## 1.1 Overview

Motion programming with textual languages is comfortable and widely used in many industries. The Kinematics Language application enables textual motion programming of kinematics. It implements a suitable subset of usable G-Code commands and provides the possibility of language extensions by the user, as well as the implementation of a completely user-defined language [\[3\]](#).

Figure 1-2 Kinematics Language



The library function blocks parse the textual program commands in a list (PathData), from where the commands are automatically executed by the function block MC\_MovePath [4](#).

Besides the possibility to define user-specific textual languages, the motion program can also be defined in G-Code. The program commands can be created manually using text editors or in case of G-Code by export of suitable (CAD) - software. The resulting program will be stored on the SIMATIC memory card. File transfer is possible by accessing the web server via the user files or using the file transfer tool.

Following benefits are provided with this library:

- comfortable motion programming using textual languages
- G-Code programming with limited instruction set
- user-defined language definition
- 4D path interpolation
- storage of textual program files on memory card
- program file transfer via web server or transfer tool
- mathematical functions for programming
- control structures and variables for programming
- HMI program editor - editing programs on HMI
- HMI program manager - load/save programs from HMI / USB

For this digital twin and the general machining use case, the base version of the Kinematics Language library has been extended with some technology-specific commands, e. g. being able to program the spindle speed using the S command (G-Code).

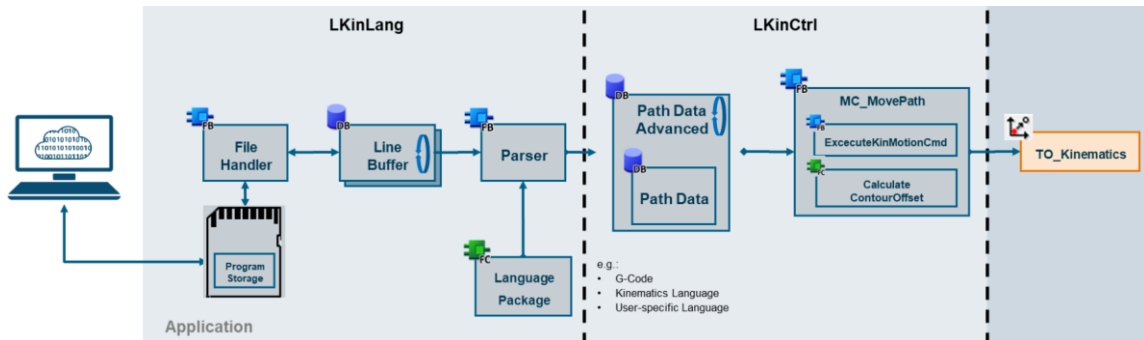
## 1.2 Mode of operation

Motion programs for kinematics can be defined in different textual languages and saved to the memory card of the PLC. File transfer is possible either via web server access to the user files or a provided file tool (SIMATIC File Transfer) transferring data to the memory card via a windows PC. The tool can be downloaded at [\3](#).

The function block (FB) File Handler enables reading the motion program files from the memory card and provides the program lines in a data block (DB) *Line Buffer* in the PLC. It is one of the main function blocks included in the kinematics language library.

Following the file handling, the FB Parser (also contained in the LKinLang library) splits the program lines to single commands. These will be interpreted according to the language definition provided in the form of a selectable Language Package. The resulting commands will be written to the PathData structure which can finally be executed with the FB MC\_MovePath from the library LKinCtrl [\4](#).

Figure 1-3 Workflow of the Kinematics Language application



## 1.3 Components used

This application example has been created with the following software components:

Table 1-1 Software components

Component	Article number
STEP 7 Professional V17.0	6ES7822-1AA07-0YA5
SIMATIC PLCSIM Advanced V4.0 <sup>1</sup>	6ES7823-1FE00-2YA6
SIMIT V10.3 Upd1	6DL8913-...30-....
NX MCD 1980	e. g. NX11113

This application example consists of the following components:

Table 1-2 Application components

Component	File name	Note
Documentation	Manual_MCD_SIMIT_Model_For_LKinLang_V1.0.pdf	
STEP 7 project	LKinLang BridgeSaw V1.0.0.zap17	
SIMIT project	LKinLang BridgeSaw V1.0.simarc	
MCD project	LKinLang BridgeSaw V1.0.0 MCD.zip	Main assembly: _Main.prt
STEP7 library	APP_LKinLang Machining_V1_0_0.zal.17	

<sup>1</sup> The correction of the PLCSIM Advanced V3.0 API is advised when using V4.0 in combination with SIMIT V10.3 Upd1 (see [\10\](#)).

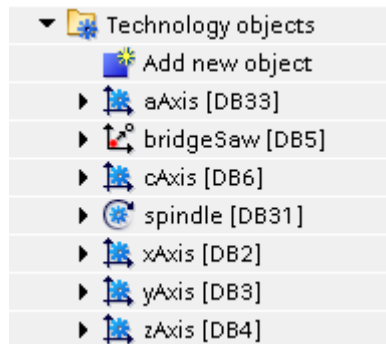
## 2 Software setup

### 2.1 TIA Portal

The TIA Portal example project contains the calls of the relevant library functions of the Kinematics Language application. Furthermore, the application specific additions are implemented in the TIA Portal and described in chapter [3.2.2](#).

The Cartesian portal bridge saw is represented by a TO\_Kinematics object in the SIMATIC S7-1500T controller. Therefore, the STEP7 project contains all required technology objects (axes) with the corresponding drives.

Figure 2-1 Technology objects in the application example



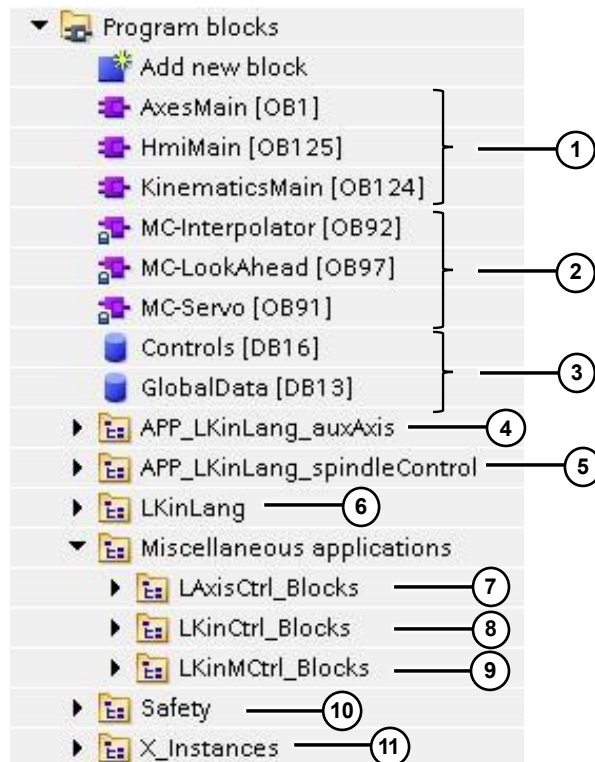
- Linear positioning axes in X, Y, and Z direction
- Rotary positioning axes in A and C direction
- Rotary speed axis as spindle
- TO\_Kinematics Cartesian Portal 3D + Orientation (X, Y, Z, and C axes)

#### NOTE

The A axis and the spindle are not contained in the TO\_Kinematics.

The application example combines the functionalities of various SIMATIC S7-1500(T) standard applications. This enlarges the possibilities to include more than the textual programming provided by the Kinematics Language (LKinLang) library. The composition of the PLC program blocks is shown in [Figure 2-2](#).

Figure 2-2 Program blocks of the digital bridge saw twin



- (1) Cyclic organizational blocks
- (2) Motion Control system blocks
- (3) Global data blocks
- (4) Auxiliary positioning via textual command (see chapter [3.1.2](#))
- (5) Spindle control via textual commands (see chapter [3.1.1](#))
- (6) Kinematics Language library [\3](#)
- (7) Standard axis control library [\5](#)
- (8) Kinematics Control library [\4](#)
- (9) Kinematics Manual Control library [\7](#)
- (10) Safety programming (utilizing [\9](#))
- (11) Function block instances

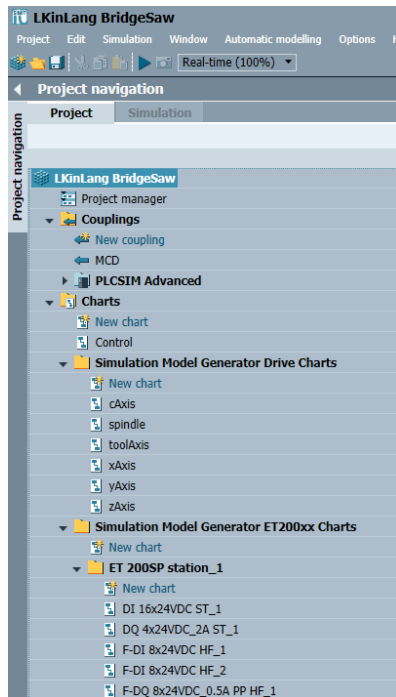
## 2.2 SIMIT

The SIMIT simulation software provides a behavior simulation of active components and periphery (e. g. drives or SIMATIC ET200SP modules). SIMIT enables the simulation of error scenarios and Safety Integrated reactions and offers the possibility of analyzing the machine behavior in a virtual environment.

The required components are organized in individual charts. The input and output signals in SIMIT are contained in couplings to both the SIMATIC PLCSIM Advanced & NX MCD model. The drive and periphery charts have been generated with the Simulation Model Generator [\6](#).



Figure 2-3 SIMIT project structure

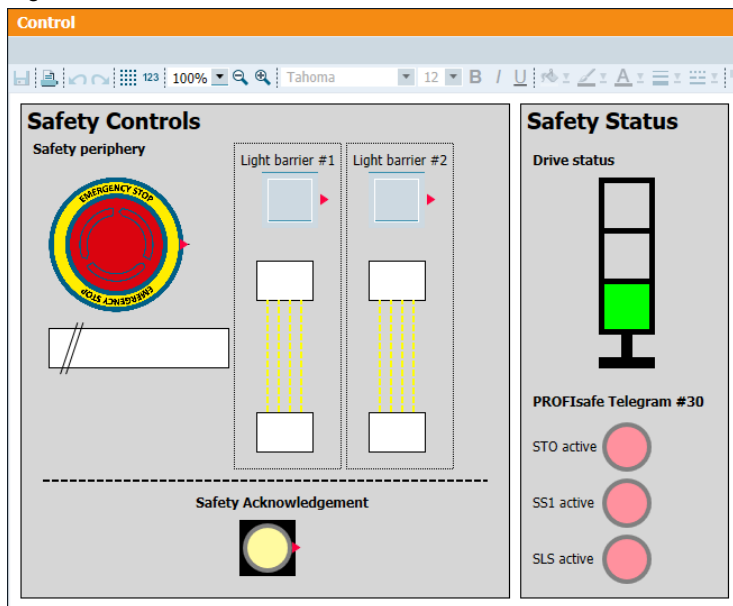


The *Control* chart and an exemplary drive chart will be explained below.

### Control chart

From the SIMIT *Control* chart, parts of the digital twin can be controlled. The *Control* chart is especially used for the simulation of external safety components, namely an emergency stop actuator and a two-stage light barrier. Additionally, there is some visual feedback about the safety status, as well as the possibility to set a safety acknowledge signal.

Figure 2-4 Control chart



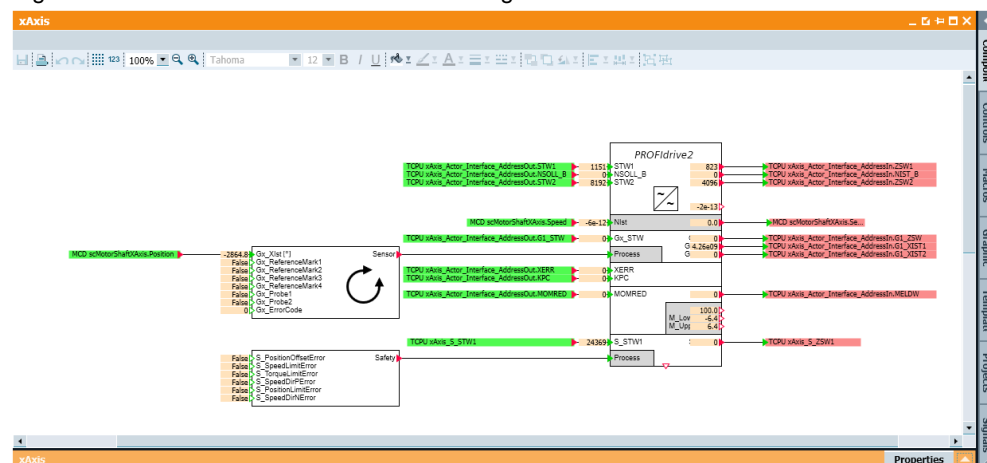
## Drive charts

For all axes, the corresponding drives are simulated in SIMIT. For the X, Y, Z, A, and C axes, SINAMICS S210 drives with PROFIdrive telegram 105 are used. A SINAMICS S120 CU310 is functioning as the spindle drive and uses the Standard telegram 2.

The telegram simulation is built with different modules:

- The *PROFdrive2* component handles the control and status words and the speed setpoint and actual speed. The control/status word is coupled to the PLCSIM Advanced instance, the speed values are linked to appropriate variables in the NX MCD coupling.
- The *Sensor* component simulates the encoder contents in the telegram. The control word, status word and increment counters are again linked to the PLCSIM Advanced instance.
- The *Process* input is fed by a *SensorProcessRotary* component. At the input of this block, the current mechanical position from the NX MCD model is evaluated and processed in an encoder format.
- The *DynamicServoControl* and *MomentumReduction* components complete the telegram contents of the PROFIdrive telegram 105.
- The *Safety30* component adds simulation of the Safety telegram 30 and is linked to the safety control word and safety status word.

Figure 2-5 xAxis chart: SIMIT PROFIdrive telegram simulation



### NOTE

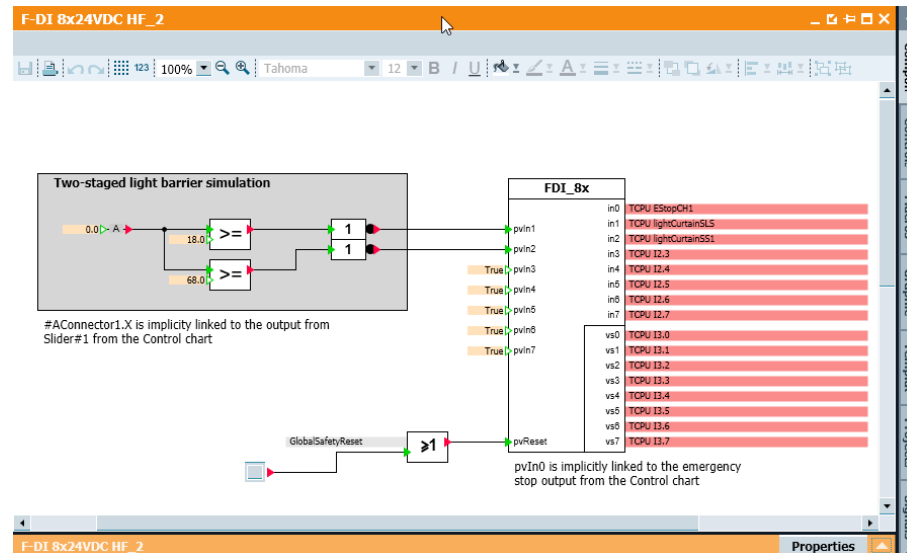
For a correct simulation with the PROFIdrive block, the parameters in SIMIT must match the parameters of the corresponding technology object in the TIA Portal. This is achieved by using the Simulation Model Generator.

If manual changes are performed, the parameters in the TIA Portal can be found under [Device] > Technology objects > [TO name] > Configuration > Basic parameters.

## ET200SP charts

As with the drive charts, the ET200SP charts have been automatically generated from the TIA Portal project using the Simulation Model Generator. The most relevant chart is containing the F-DI module simulation for the evaluation of the emergency stop and light barrier signals.

Figure 2-6 ET200SP peripheral chart



## 2.3 NX Mechatronics Concept Designer

NX MCD allows the simulation and test of mechanical machine components in a virtual setting. It features the combination of CAD machine construction data with a physical simulation engine. The kinematic capabilities of an NX MCD model are defined by different mechanical components.

These components include rigid bodies and mechanical joints and are configured in the physics navigator of MCD. Rigid bodies enable the movement of machine components, as they are subject to the physics simulation in simulation mode. By linking multiple rigid bodies with suitable mechanical joints (e. g. sliding joint, hinge joint), the machine kinematic chain is implemented. For controlled movements, speed controllers are created and linked to the corresponding joints.

By linking the setpoint of these controllers via SIMIT to the PLCSIM Advanced instance, the SiL toolchain is completed.

This process of equipping the virtual machine with physical properties is referred to as kinematization.

**NOTE** This application example contains a sufficiently kinematized MCD model of a bride saw.

[Figure 2-7](#) shows the contents of the Physics Navigator for the virtual portal kinematic. The basic moving components are defined as rigid body (e. g. *rbMotorShaftZAxis\_1*). An appropriate joint is assigned to the rigid bodies (e. g. a rotary hinge joint acting on the motor shaft *hjMotorShaftZAxis*). Finally, a speed controller is connected to the joints (e. g. *scMotorShaftZAxis*). The input and output values of the speed controllers are the interfacing variables towards SIMIT. The translation from the Z axis motor shaft rotation to a linear Z axis movement is established by the screw joint *scjMotorShaftZAxis*, with a configurable screw pitch.

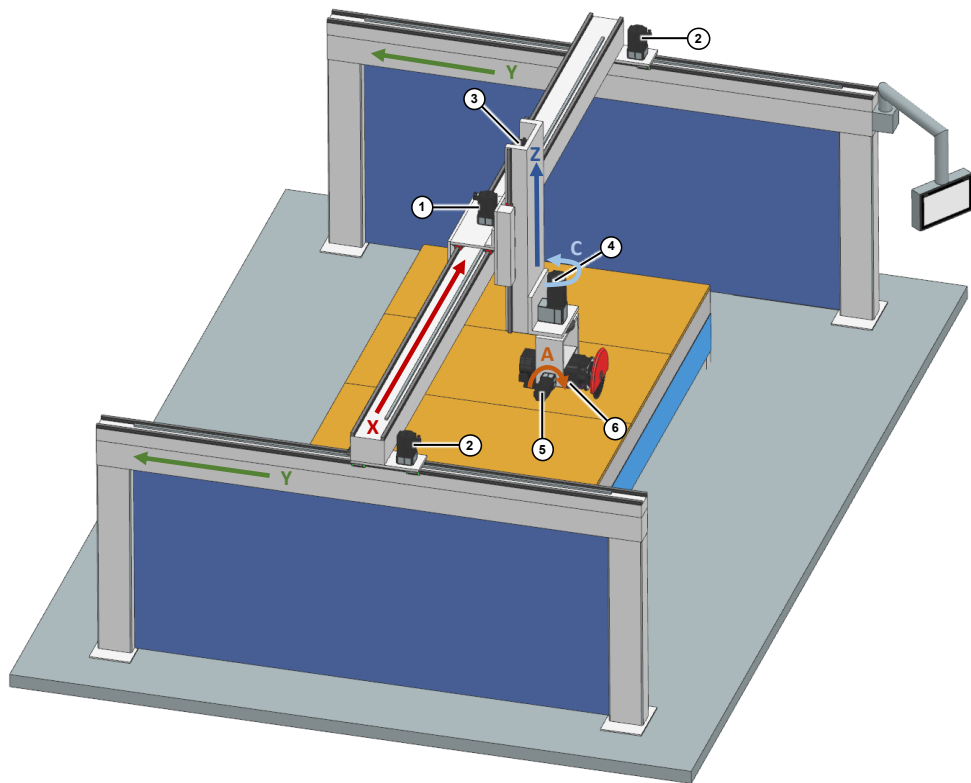
Figure 2-7 NX MCD elements from Physics Navigator

Basic Physics		Joints and Constraints		Sensors and Actuators	
<input checked="" type="checkbox"/>	rbMotorShaftAAxis_1	Rigid Body	<input checked="" type="checkbox"/>	fjNozzleFlange	Fixed Joint
<input checked="" type="checkbox"/>	rbMotorShaftXAxis_1	Rigid Body	<input checked="" type="checkbox"/>	fjOrientToolAxis	Fixed Joint
<input checked="" type="checkbox"/>	rbMotorShaftY1_1	Rigid Body	<input checked="" type="checkbox"/>	hjMotorShaftCAxis	Hinge Joint
<input checked="" type="checkbox"/>	rbMotorShaftY2_1	Rigid Body	<input checked="" type="checkbox"/>	hjMotorShaftXAxis_1	Hinge Joint
<input checked="" type="checkbox"/>	rbMotorShaftZAxis_1	Rigid Body	<input checked="" type="checkbox"/>	hjMotorShaftY1	Hinge Joint
<input checked="" type="checkbox"/>	rbNozzle	Rigid Body	<input checked="" type="checkbox"/>	hjMotorShaftY2	Hinge Joint
<input checked="" type="checkbox"/>	rbShaftOrientAxis	Rigid Body	<input checked="" type="checkbox"/>	hjMotorShaftZAxis	Hinge Joint
<input checked="" type="checkbox"/>	rbSpindle_1	Rigid Body	<input checked="" type="checkbox"/>	hjSpindle	Hinge Joint
<input checked="" type="checkbox"/>	rbToolAxis_1	Rigid Body	<input checked="" type="checkbox"/>	hjToolOrientationAxis	Hinge Joint
<input checked="" type="checkbox"/>	rbXAxis_1	Rigid Body	<input checked="" type="checkbox"/>	scjMotorShaftZAxis	Screw Joint
<input checked="" type="checkbox"/>	rbYAxis_2	Rigid Body	<input checked="" type="checkbox"/>	sjXAxis	Sliding Joint
<input checked="" type="checkbox"/>	rbZAxis_1	Rigid Body	<input checked="" type="checkbox"/>	sjYAxis	Sliding Joint
			<input checked="" type="checkbox"/>	sjZAxis	Sliding Joint
			<input checked="" type="checkbox"/>	INTERNAL_scXAxis	Speed Control
			<input checked="" type="checkbox"/>	INTERNAL_scYAxis	Speed Control
			<input checked="" type="checkbox"/>	scMotorShaftCAxis	Speed Control
			<input checked="" type="checkbox"/>	scMotorShaftXAxis	Speed Control
			<input checked="" type="checkbox"/>	scMotorShaftY1	Speed Control
			<input checked="" type="checkbox"/>	scMotorShaftZAxis	Speed Control
			<input checked="" type="checkbox"/>	scSpindle	Speed Control
			<input checked="" type="checkbox"/>	scToolOrientationAxis	Speed Control

### 2.3.1 Overview

In the MCD project, a Cartesian portal of a bridge saw is prepared.

Figure 2-8 MCD model overview



- (1) X axis motor
- (2) Y axis motors
- (3) Z axis motor
- (4) C axis motor
- (5) A axis motor
- (6) Spindle

## 3 Engineering

The application Kinematics Language serves as the base for the digital bridge saw twin. Inherent to this application is the possibility of extending the functionality of its textual programming with user-defined commands.

For the bridge saw example, this mechanism is used to implement a spindle speed programming command with the S word, commonly used in abrasive machines running G-Code.

### 3.1 Description of interface

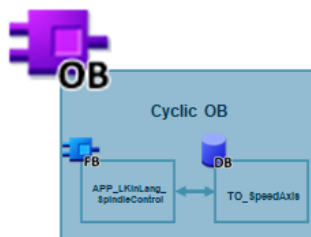
#### 3.1.1 APP\_LKinLang\_SpindleControl

##### Principle of operation

The function block APP\_LKinLang\_SpindleControl can be used to control an axis of type TO\_SpeedAxis in the context of a speed-controlled spindle for machining purposes. The block inputs refer to the typical G-Code commands for spindle control, namely M3 (clockwise operation), M4 (counter-clockwise operation) and M5 (spindle stop).

Control of rotary tools/spindles via M commands is widely used in many G-Code applications but currently not included in the standard LKinLang library functionality. The function block APP\_LKinLang\_SpindleControl acts as an extension to the core functionality.

Figure 3-1 Call and data exchange of the spindle control function block



The use of machine function (M functions) is part of the provided G-Code capabilities of the Kinematics Language application. Therefore, the M3, M4, and M5 inputs need to be linked to the corresponding structure containing the flag status information. In the example project, the flag values are contained in the "GlobalData".booleanFlagArray[n] array, where  $n$  signifies the M function number.

To provide a useful functionality, a way of setting the desired spindle speed setpoint from the G-Code program is necessary. A typical G-Code line can look as follows:

```
S2000 M3
```

This encodes the request of setting a speed setpoint of 2000 rpm and starting the spindle in clockwise direction.

Once the spindle is already running, a newly programmed S word requires an update of the speed setpoint.

```
S=750
```

The support for S word programming is implemented using the user-defined G-Code capabilities. A detailed description of the general procedure is described in [3](#) (Chapters “User-defined G-Code” and “User-defined language”).

```


=====
// S function
//
=====
// Use Sxxx to program the spindle speed
//
=====
// Parameter name definition
#parameterDefinition.parameterNameString := 'S';
#parameterDefinition.parameterNameNumber := -1;

#parameterDefinition.directValues.valueFlags[1].flag := 0; // first value flag for spindle speed
#parameterDefinition.followingValues[1].valueFlags.value := TRUE;

// Matching to PathData
#parameterDefinition.priorities.valueFlags.flag := "LKINLANG_PRIO_DEFINED";
#parameterDefinition.priorities.valueFlags.value := "LKINLANG_PRIO_DEFINED";

// Valid counts per line
#parameterDefinition.validCountPerLine := 1;
=====
    
```

The configuration results in the transmission of the programmed spindle speed into the “GlobalData”.valueFlagArray[0].

 <b>CAUTION</b>	<p><b>Spindle speed can be changed without programming S command</b></p> <p>The valueFlagArray – which is used to transfer the desired speed setpoint from G-Code program to the controller – can also be programmed with an H command. This can result in unexpected changes to the spindle speed.</p> <p>Exclusively use the valueFlagArray[0] for the S command and avoid using the H0 command!</p>
---	--

**Interface**

Figure 3-2: APP\_LKinLang\_SpindleControl

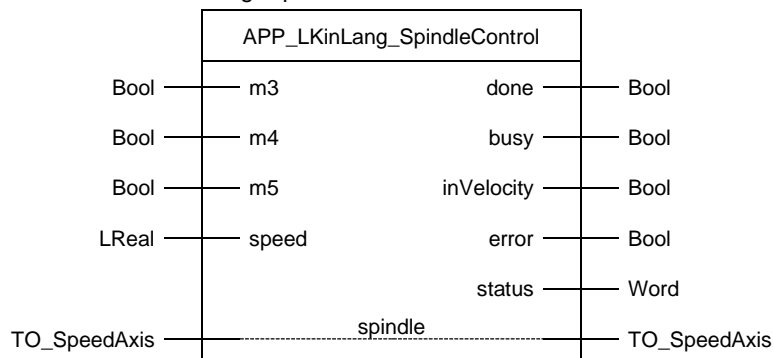


Table 3-1: Parameter of APP\_LKinLang\_SpindleControl

Name	P-Type	Data Type	Comment
m3	IN	Bool	Start spindle in clockwise direction
m4	IN	Bool	Start spindle in counter-clockwise direction
m5	IN	Bool	Stop spindle
speed	IN	LReal	Spindle speed setpoint value
done	OUT	Bool	TRUE: Commanded spindle halt has been completed successfully
busy	OUT	Bool	TRUE: FB is not finished
inVelocity	OUT	Bool	TRUE: Spindle has reached speed setpoint
error	OUT	Bool	TRUE: An error occurred during the FB execution
status	OUT	Word	16#0000 - 16#7FFF: Status of the FB, 16#8000 - 16#FFFF: Error identification
spindle	IN_OUT	TO_SpeedAxis	

### 3.1.2 APP\_LKinLang\_auxiliaryAxis

#### Principle of operation

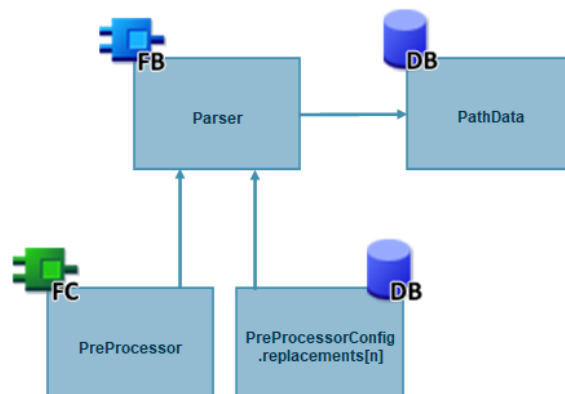
In many machining operations there is a need for auxiliary and supporting axes in addition to the interpolating axis group. The tasks performed by these auxiliary axes can stem from a wide range of uses and contain examples like workpiece clamping, material transport, or static tool orientation.

The nature of these tasks does not require a strict interpolation, but a coordination and integration into the textual (G-Code) programming is necessary. As an example, performing a tool orientation or clamping action is required before a sensible machining operation can safely be performed.

The Kinematics Language application is based on the technology object TO\_Kinematics. This technology object has a limitation regarding its connectable axes. Thereby, there is also an implicit limit to the number of axes that can be addressed by the textual programming.

In this application example, the digital twin of the bridge saw contains a larger number of axes than a single TO\_Kinematics can handle. The tool orientation axis (rotating around the X axis – therefore conventionally referred to as an A axis) is its own technology object (TO\_PositioningAxis). It will serve as an example of how to address and integrate an additional axis into the textual program.

Figure 3-3 Data exchange and composition of the textual replacements



The Kinematics Language application features a preprocessor function that can be used for replacing definable parts of the textual program. It is configured via the input of type "LKinLang\_typePreProcessorReplacement" at the LKinLang\_Parser function block.

The replacement mechanism is used applicative to adapt a textual command for an axis positioning job into a LKinLang compatible combination of an M function (for triggering) and an H function (for the value transfer).

The following textual command shall be used to position a specific auxiliary axis.

```
POS [auxAx1] = 350.0
```

The following replacement by the preprocessor is configured:

Figure 3-4 preProcessorConfig for auxiliary axis programming

preProcessorConfig	"LKinLang_typePreProcessorReplacement"	
replacements	Array[1.."LKINLANG_NUMBER_OF_REPLACEMENTS"] ...	
replacements[1]	Struct	
searchTerm	String["LKINLANG_MAX_NUMBER_OF_TOKEN_PER_LI..."]	'POS[auxAx1]'
replaceTerm	String["LKINLANG_MAX_NUMBER_OF_TOKEN_PER_LI..."]	'M1 H1'

This results in an effective textual command for further parsing.

```
M1 H1 = 350.0
```

Here, only implemented G-Code functions are used and the parsing can be performed successfully. When the respective line is reached in the G-Code program, the parser raises the flag in "GlobalData".booleanFlagArray[1] and provides the programmed position setpoint in "GlobalData".valueFlagArray[1].

A positioning axis can subsequently be supplied with this information. For this purpose, the application SIMATIC S7-1500T Standard application axis control (LAxisCtrl) is utilized [5](#). This reduces the user's effort and allows for a very quick and reliable axis integration.

**NOTE**

The APP\_LKinLang\_auxiliaryAxis implementation is a simple showcase of how to use the flexibility of the Kinematics Language application for a custom task.

**Interface**

Figure 3-5: APP\_LKinLang\_auxiliaryAxis

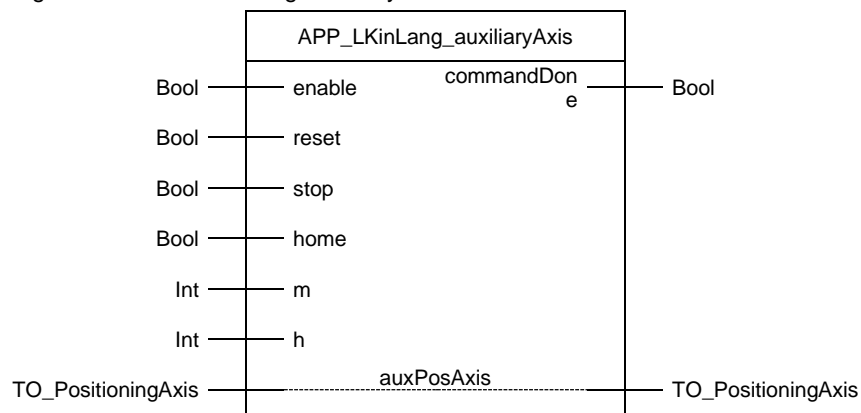




Table 3-2: Parameter of APP\_LKinLang\_auxiliaryAxis

Name	P-Type	Data Type	Comment
enable	IN	Bool	TRUE: enable LAxisCtrl FB and technology object
reset	IN	Bool	Rising edge: Acknowledgement of technology alarms or restart of the axis (depending on configuration)
stop	IN	Bool	Rising edge: Brake an axis until it comes to a standstill. Note: MC_Halt is triggered internally
home	IN	Bool	Rising edge: Home axis
m	IN	Int	Specify M function from replacement to trigger positioning job
h	IN	Int	Specify H function from replacement to transfer target position
commandDone	OUT	Bool	TRUE: The selected basic motion command has completed without error
auxPosAxis	IN_OUT	TO_PositioningAxis	reference to the technology object

### 3.1.3 Relevance of FlagModes

Both the implementation of the S command capability as well as the auxiliary axis support are utilizing M and H functions. Especially the Boolean M functions can be further configured by selecting a suitable flag mode. This can lead to very different behaviors, which need to be chosen depending on the specific machine requirements.

A list of the available flag mode is shown in [Table 3-3](#).

Table 3-3 Available flag modes

Flag Mode	
0	Flag deactivated
1	SET_BEFORE_AND_NO_RESET
2	SET_BEFORE_AND_RESET_AFETR
3	SET_BEFORE_AND_RESET_AFTER_ONE_CYCLE
5	SET_BEFORE_AND_NO_RESET_AND_WAIT_FOR_ACKNOWLEDGE
10	SET_IN_REMAINING_DISTANCE_TO_TARGET
11	SET_AFTER_AND_NO_RESET
13	SET_AFTER_AND_RESET_AFTER_ONE_CYCLE
15	SET_AFTER_AND_NO_RESET_AND_WAIT_FOR_ACKNOWLEDGE

To exemplify the marked difference the flag mode can have on an operation, the same G-Code program is used in two cases where only the flag mode of the M function M1 is altered. The implementation of the auxiliary axis using the preprocessor replacement and the M1 command is assumed.

```
G0 X100 Y200 Z150
G1 Z50 F100.0
POS[auxAx1]=20.0
G1 Z20
```

### **FlagMode = SET\_BEFORE\_AND\_RESET\_AFTER**

With this flag mode, the TO\_Kinematics tool center point (TCP) is positioned first to the point with the coordinates X=100, Y=200, and Z=150 with rapid movement. After finishing the first step, the Z axis is lowered to position 50.0.

Then, the auxiliary axis positioning is triggered with the target destination at 20.0. Since the M function is not requiring a reset, the next program step is immediately executed and the Z axis is lowered even further. This Z axis movement is occurring simultaneously (but not interpolated) to the auxiliary axis movement.

### **FlagMode = SET\_AFTER\_AND\_NO\_RESET\_AND\_WAIT\_FOR\_ACKNOWLEDGE**

As before, the TO\_Kinematics TCP is positioned first to X=100, Y=200, and Z=150 with rapid movement and after finishing the first step, the Z axis is lowered to position 50.0.

Again, the auxiliary axis positioning is triggered with the target destination at 20.0. Since the M function is now waiting until a dedicated acknowledgment is performed, the next program step is not executed and the Z axis is staying at the position 50.0. Only when the auxiliary axis has reached its target position and the job has been acknowledged can the program execution continue with the lowering of the Z axis.

This shows how the M function flag mode can be used to define the sequential behavior.

## **3.2 Project integration**

### **3.2.1 Requirements**

To run the language extension for machining centers with textual programming, a TIA Portal V17 project needs to be setup. It requires a suitable TO\_Kinematics and a suitable number of TO\_PositioningAxis. In addition to the kinematic (which is composed of underlying positioning axes), a speed axis for the spindle and possible further positioning axes for auxiliary purposes are necessary.

All axes and the kinematic must be configured and commissioned and a first test run via the TIA-built-in control panel is advised.

#### **NOTE**

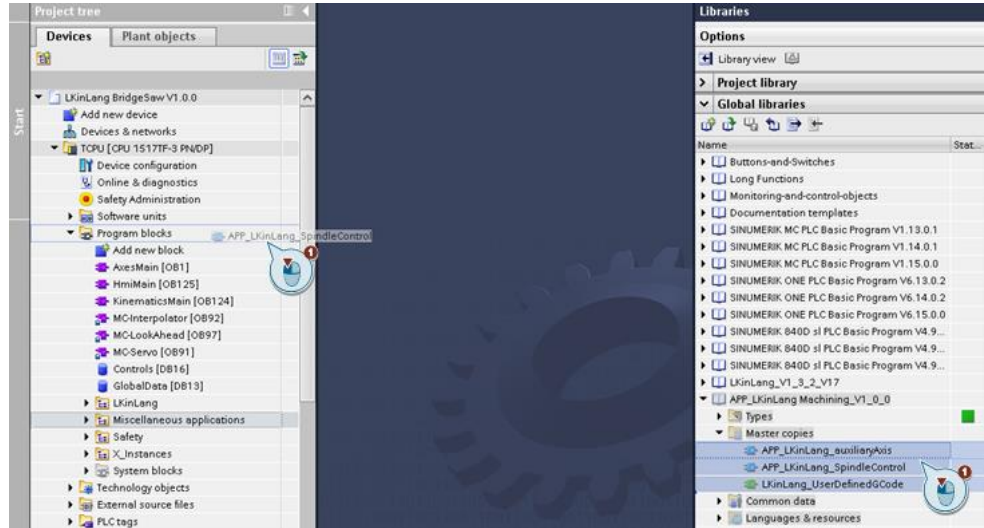
The example project for the fundamental Kinematics Language application can serve as a good starting point for the functional additions for machining operations. The example project (for TIA V16) can be downloaded under [13](#).

The TIA project for the digital twin comes with fully prepared technology objects.

### 3.2.2 Import of application

The function and function blocks containing the language extension and spindle control / positioning control can be inserted into the PLC project via drag-and-drop (see [Figure 3-6](#)).

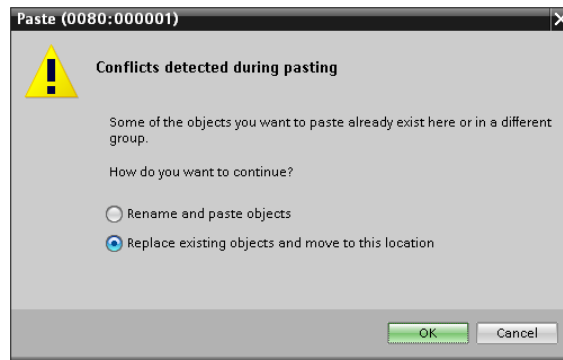
Figure 3-6 Integration the application library blocks into TIA Portal



It is assumed, that the LKinLang library has been imported beforehand.

In this case, the function LKinLang\_UserDefinedGCode will already be present in the project and the dialog from [Figure 3-7](#) will occur.

Figure 3-7 Chose pasting option for duplicate blocks



Selecting the option *Replace existing objects and move to this location* will overwrite any previous alterations in the *LKinLang\_UserDefinedGCode* function. Select this option if you have not changed any language definitions and want to use the S command capability of this application example.

Should you have already done user defined adaptations to the LKinLang, select the *Rename and past objects* option. Now you can merge your existing language extensions with the ones included in the application example.

The FB APP\_LKinLang\_SpindleControl does not require the import of additional PLC data types or PLC tags. Only the respective spindle control function block needs to be imported from the library (see [Figure 3-6](#)), if the auxiliary axis function will not be used.

The FB APP\_LKinLang\_auxiliaryAxis uses the LAxisCtrl\_PosAxis function block from the LAxisCtrl library. Therefore, this library with its PLC data types and PLC blocks needs to be imported. The process is described in [15](#).

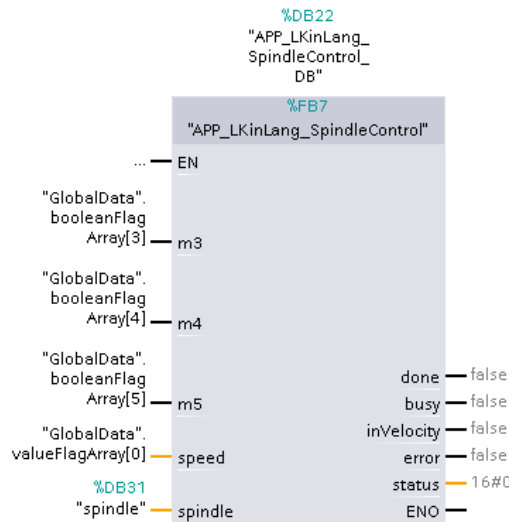
### 3.3 Operation

#### 3.3.1 APP\_LKinLang\_SpindleControl

##### Interface controls

The FB APP\_LKinLang\_SpindleControl is triggered via rising edges on the input signals *m3*, *m4*, and *m5*. The names of these inputs are derived from the standardized machine functions (M functions) for spindle control in G-Code textual programming.

Figure 3-8 Call of SpindleControl FB from the cyclic OB



A rising edge on the *m3* input triggers the clockwise spindle operation with the speed specified at the *speed* input.

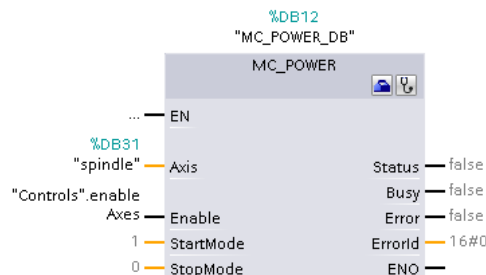
A rising edge on the *m4* input triggers the counterclockwise spindle operation with the speed specified at the *speed* input.

A rising edge on the *m5* input triggers a halt command for the spindle.

If multiple inputs receive a rising edge simultaneously, the halting command takes priority over the clockwise and counterclockwise rotation commands.

Enabling the speed axis technology object which is connected as an INOUT parameter to the SpindleControl needs to be handled outside of the block, using the MC\_POWER system function.

Figure 3-9 General technology object control with Motion Control commands



Resetting of technology object errors requires external handling as well.

## Diagnostics & Status

The diagnostic output information is supplied in the form of the PLCOpen based outputs *done*, *busy*, *error*, and *status*. Additionally, the block signals the reaching of the request velocity with the *inVelocity* output (see [Figure 3-2](#)).

### 3.3.2 APP\_LKinLang\_auxiliaryAxis

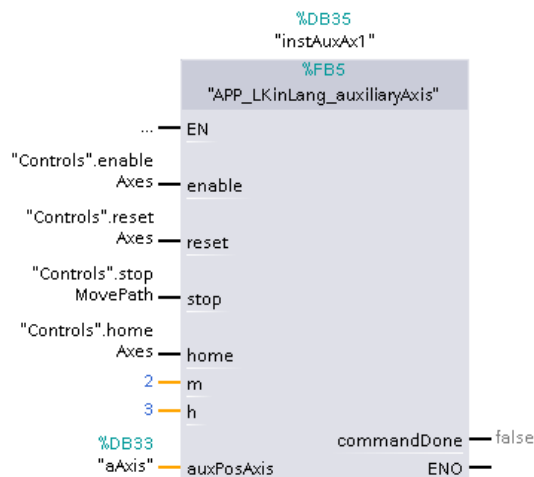
#### Interface controls

The FB APP\_LKinLang\_auxiliaryAxis is an enveloping function block for an instance of the LAxisCtrl\_PosAxis FB [5](#). The interface provides the signals *enable*, *reset*, *stop*, and *home* for the general operation of a positioning axis technology object.

Furthermore, the *m* and *h* inputs are used for the configuration of the LKinLang Boolean and value function numbers. This is corresponding directly with the chosen string replacements in the LKinLang\_Parser as shown in chapter [3.1.2](#).

For example, if the desired textual positioning command (`pos[auxAx1] = 123.45`) is translated into an intermediate line utilizing M2 and H3 (`M2 H3=123.45`) then the *m* and *h* inputs of the APP\_LKinLang\_auxiliaryAxis require the assignment of 2 and 3 respectively.

Figure 3-10 Call of auxiliary axis FB from cyclic OB



The *m* and *h* inputs do NOT operate with an enable/execute logic. Rather, the concrete triggering of the M and H function is performed within the FB by assessing the *"GlobalData".booleanFlagArray[#m]* and *"GlobalData".valueFlagArray[#h]*.

#### Diagnostic & Status

The APP\_LKinLang\_auxiliaryAxis FB signals a completed positioning task with a *commandDone* bit (see [Figure 3-5](#)). Further diagnostic information is possible by accessing the LAxisCtrl\_PosAxis multi-instance inside the auxiliaryAxis instance data block.

## 3.4 Error handling

### 3.4.1 APP\_LKinLang\_SpindleControl – Error identifiers

[Table 3-4](#) shows the error identifiers of the *APP\_LKinLang\_SpindleControl*.

Table 3-4 Error identifiers – APP\_LKinLang\_SpindleControl

Name	Type	Value	Comment
ERR_MOVE_VELOCITY	Word	16#8000	Error on execution of internal MC_MOVEVELOCITY
ERR_HALT	Word	16#8001	Error on execution of internal MC_HALT
ERR_UNDEFINED_STATE	Word	16#8600	Internal error code

### 3.4.2 APP\_LKinLang\_SpindleControl – Error reaction and acknowledgement

#### Error reaction

If an error occurs during execution of the FB (output *error* = TRUE), the *status* output shows the error number (identifier). The *error* bit is TRUE for at least one cycle.

#### Acknowledgement

Errors will be automatically acknowledged after one cycle when the inputs M3, M4, and M5 are set to FALSE.

### 3.4.3 APP\_LKinLang\_auxAxis

The implementation of the auxiliary axis positioning within the LKinLang framework relies on the use of the LAxisCtrl library. There axis control by this library is not altered, and the integration into the textual language interpretation is handled via M and H function.

The user is referred to the LAxisCtrl diagnostics and error identifiers for the diagnosis of the auxiliary axis positioning status [5](#).

## 4 Operating the Digital Twin

### 4.1 Starting the simulation

The startup of the simulation requires a certain sequence. The recommended procedure is:

1. Prepare NX MCD model
2. Open SIMIT project and start simulation
3. Optional: Download project to PLCSIM Advanced instance
4. Start WinCC HMI

The individual steps are described in more detail in the following chapters.

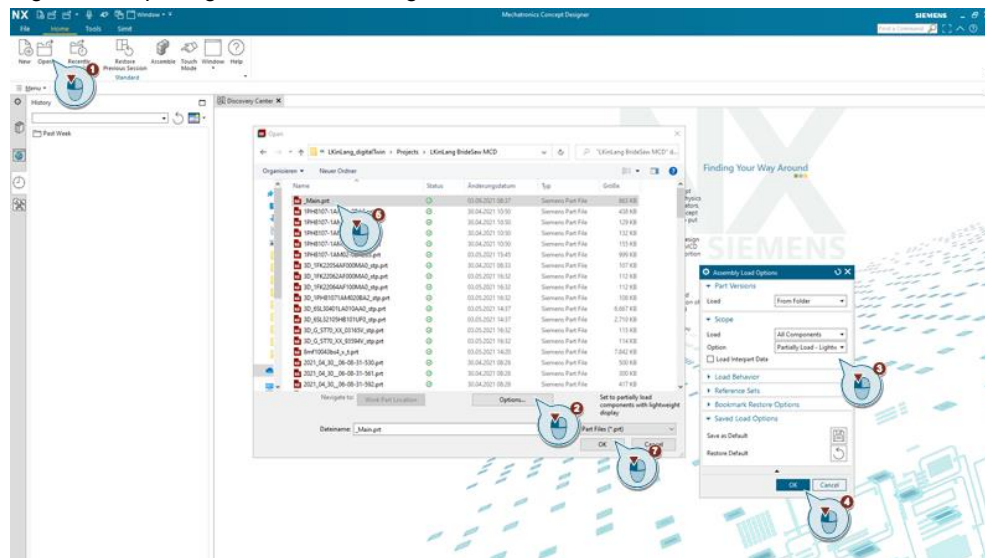
Also note a potentially necessary initial error acknowledgement (see chapter [4.3.1](#)).

#### 4.1.1 NX MCD

The NX MCD model provided with this application example must be opened. Furthermore, it is required to enable a load option of at least “Partially Load – Lightweight Display”

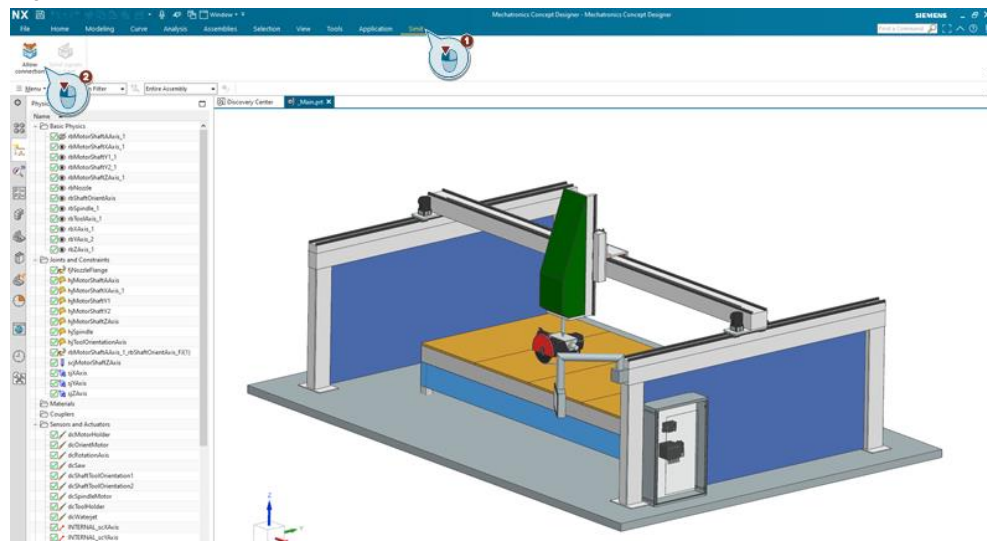
The full model is loaded by selecting the `_Main.prt` file. This contains the assembly information for all necessary parts, joints, and controllers.

Figure 4-1 Opening the NX MCD bridge saw model



In addition to opening the NX MCD model, the connectivity between NX MCD and SIMIT needs to be allowed. This is achieved via the SIMIT toolbar in NX MCD (see [Figure 4-2](#)).

Figure 4-2 Allow SIMIT ↔ NX MCD connection

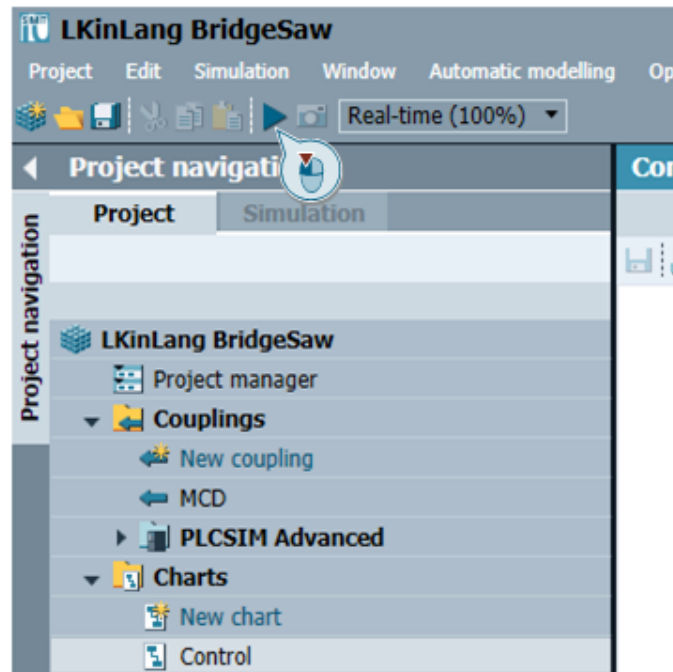


### 4.1.2 SIMIT

The simulation is started from SIMIT, which will also handle the startup of the configured PLCSIM Advanced instance and the start of the NX MCD simulation mode.

**NOTE** When the simulation is started for the first time in a new environment, the PLCSIM Advance instance will not contain any project information. Therefore, the instance will be in STOP mode. The TIA Portal project can be downloaded.

Figure 4-3 Simulation start in SIMIT



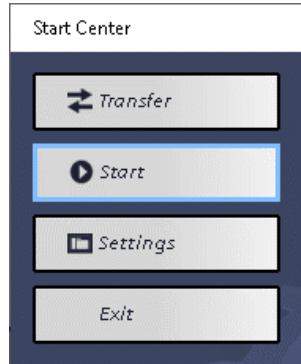


### 4.1.3 WinCC Runtime

An exemplary WinCC HMI featuring the LKinLang faceplates (*ProgramEditor*, *ProgramManager*) and some additional functionality is provided with the TIA Portal example project.

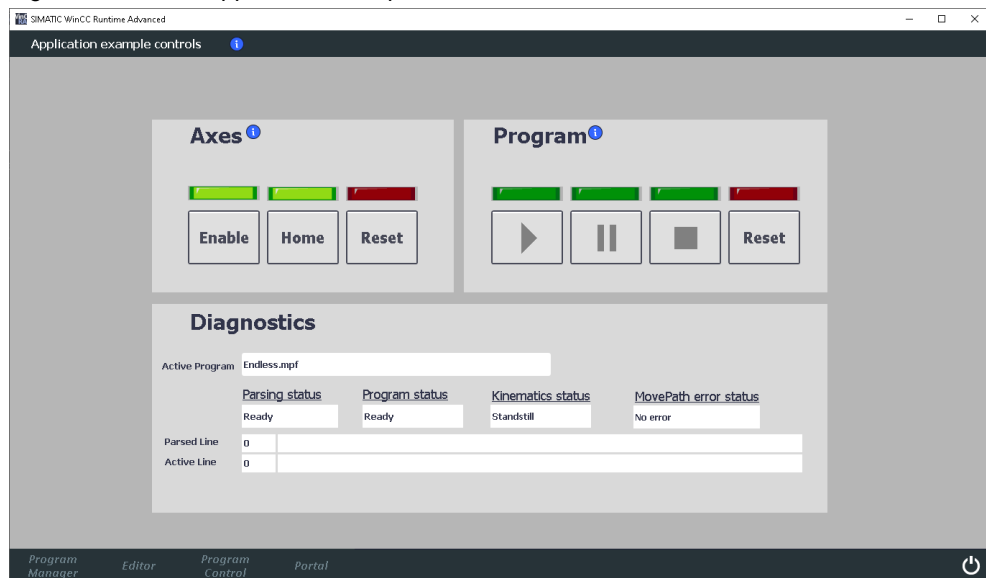
The HMI runtime can be started from the TIA Portal or via the WinCC Runtime Loader.

Figure 4-4 WinCC Runtime Loader



If the WinCC has never been loaded/started from the TIA Portal, the Transfer mode of the WinCC Runtime Loader needs to be selected. Then, a download from TIA Portal is possible.

Figure 4-5 WinCC application example screen



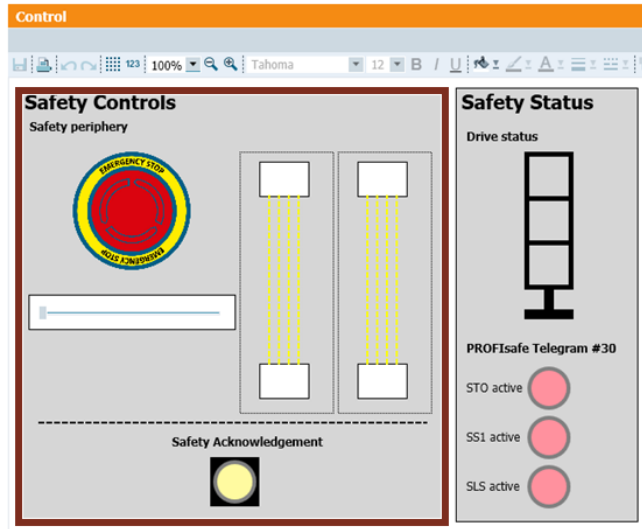
The detailed possible interactions with the digital bridge saw twin via the HMI are described in the following section.

## 4.2 Operating the simulation

The digital twin relies on two main application parts for operation.

1. The simulation of safety acknowledgment and safety periphery (emergency stop, light barrier) is controlled from the Control diagram in SIMIT.

Figure 4-6 SIMIT Control diagram with safety controls



2. All other controls and interactions with the digital twin are carried out through a WinCC Advanced HMI (included in the TIA Portal project, see [Figure 4-5](#)), or through TIA Portal directly.

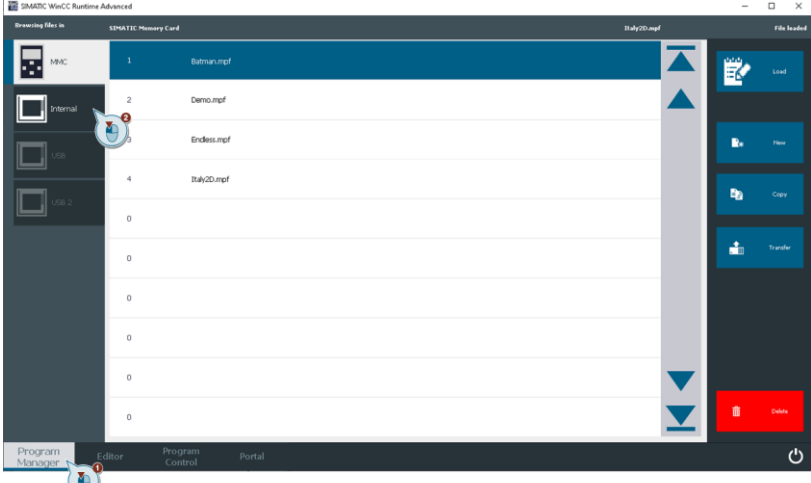
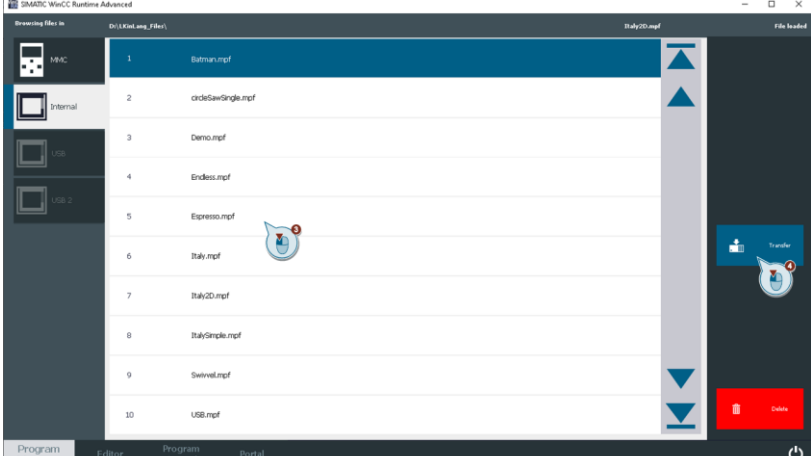
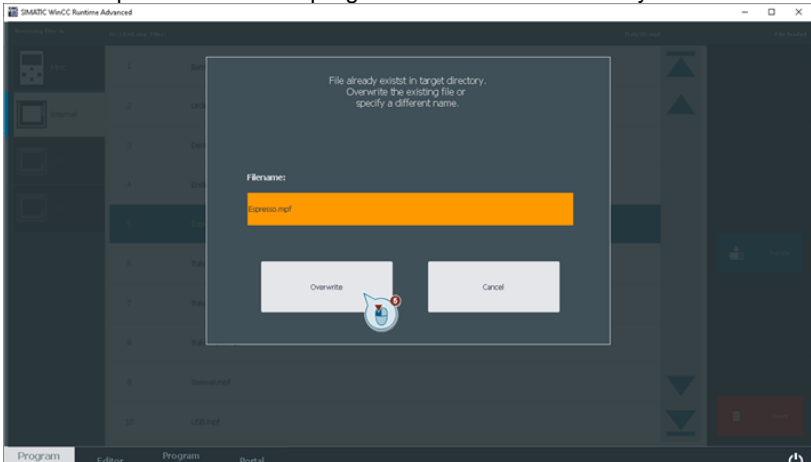
Figure 4-7 Accessing Control DB trough TIA Portal (online)

Name	Data type	Start value	Monitor val.
Static			
enableAxes	Bool	false	TRUE
axesEnabled	Bool	false	TRUE
resetAxes	Bool	false	FALSE
axesError	Bool	false	FALSE
homeAxes	Bool	false	FALSE
axesHomed	Bool	false	FALSE
homePos	Array[1..4] of LReal		
homeMode	Array[1..4] of Int		
restartMode	Array[1..5] of Bool		
startMode	Array[1..4] of DInt		
stopMode	Array[1..4] of Int		
loadFile	Bool	false	FALSE
saveFile	Bool	false	FALSE
stopFileHandler	Bool	false	FALSE
executeParser	Bool	false	FALSE
stopParser	Bool	false	FALSE
executeMovePath	Bool	false	FALSE
interruptMovePath	Bool	false	FALSE
stopMovePath	Bool	false	FALSE

### 4.2.1 Program Management – File Transfer

[Table 4-1](#) shows the procedure for transferring a textual part program to the SIMATIC memory card. A folder on the computer executing the WinCC runtime serves as a source for the file transfer, the preconfigured address is *D:\KinLang\_Files\*. If such a directory does not exist, the application will create it.

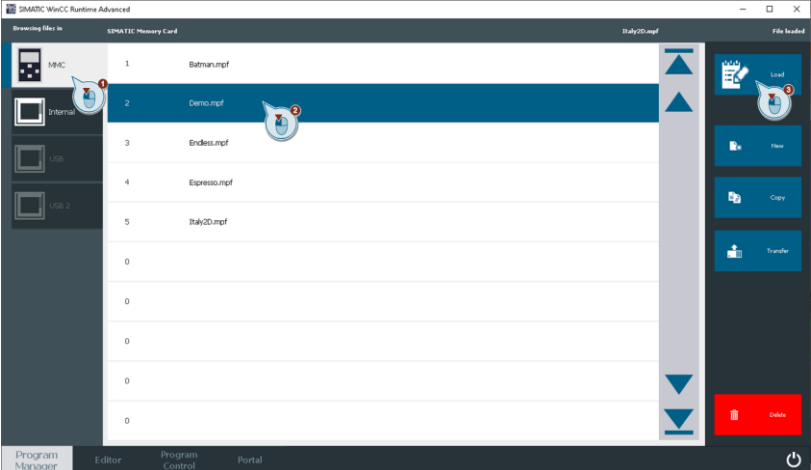
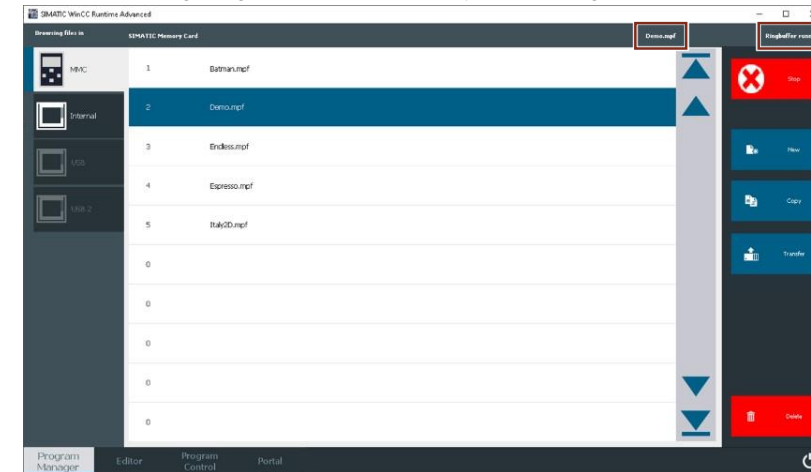
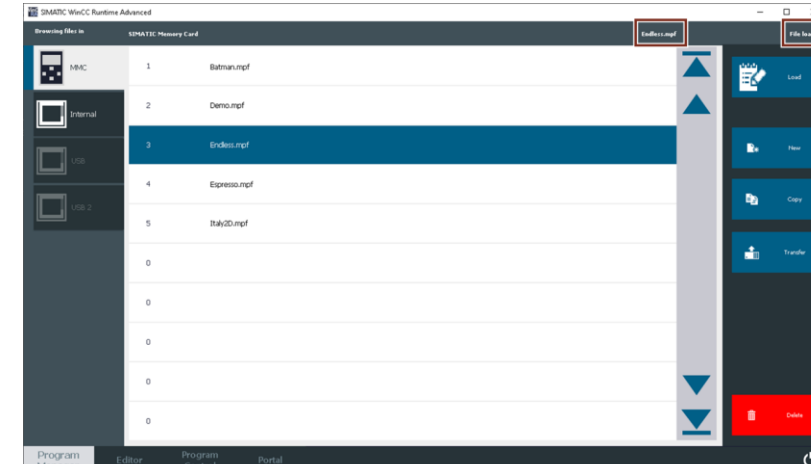
Table 4-1 Program transfer in WinCC

No.	Action
1.	
2.	
3.	<p data-bbox="598 1496 1220 1529">Optional: Overwrite program on SD card if it already exists</p> 

### 4.2.2 Program Management - Selecting a program

Selecting a program for execution is performed in the Program Manager faceplate according to the step in [Table 4-2](#).

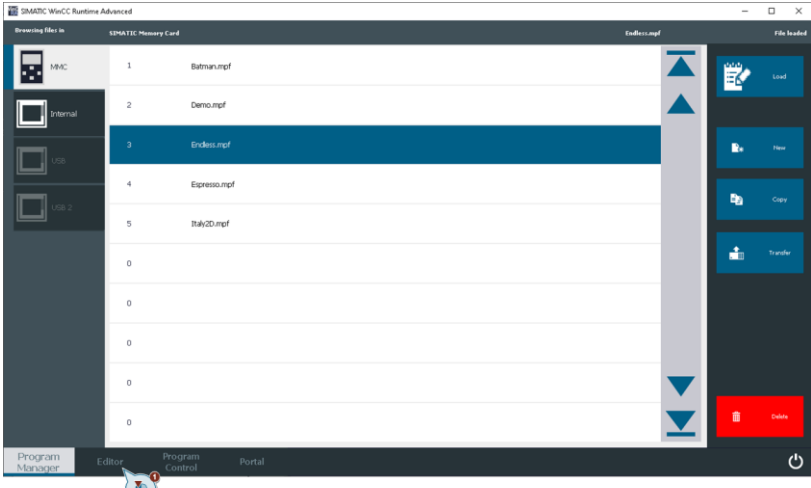
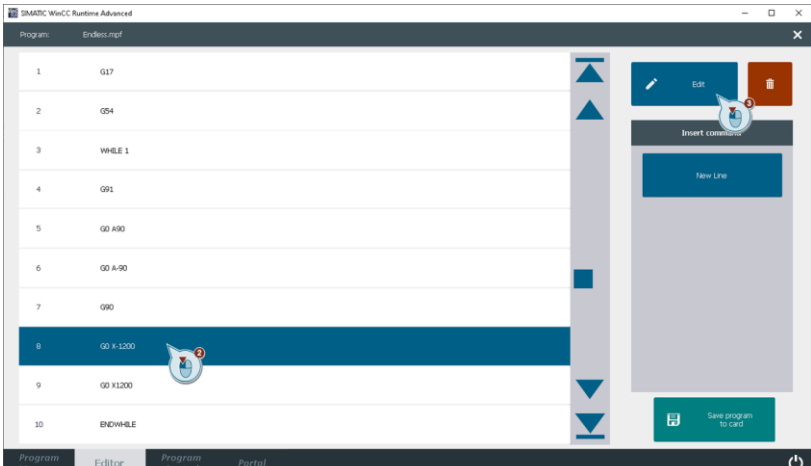
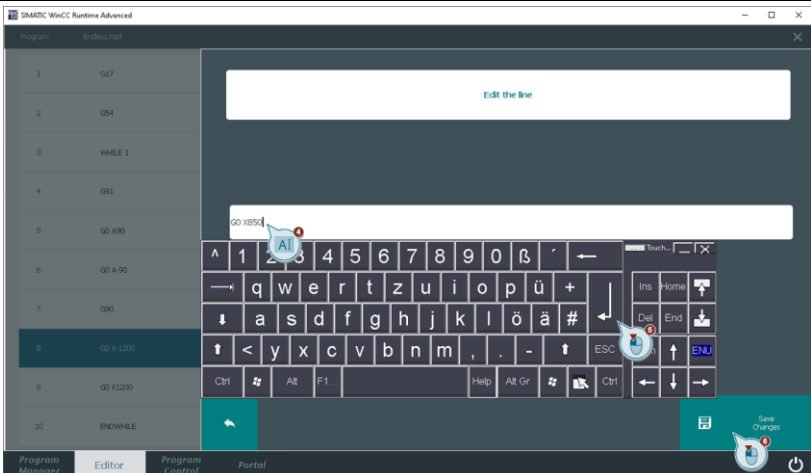
Table 4-2: Program selection in WinCC

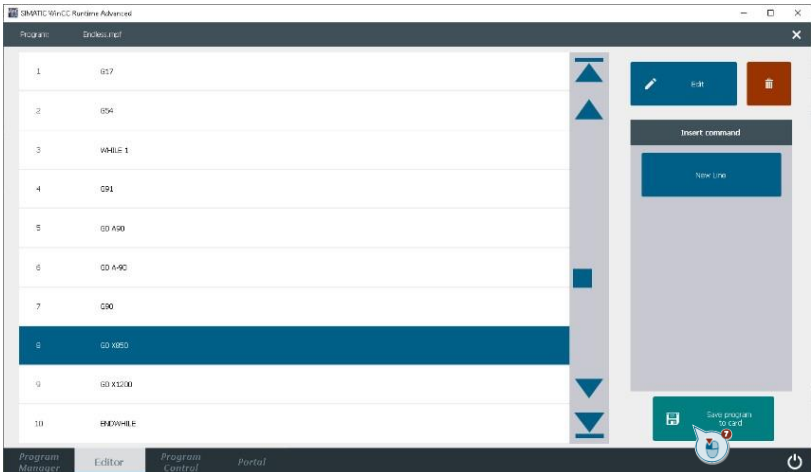
No.	Action
1.	
2.	<p data-bbox="612 967 1222 996">a A long program will automatically start in ringbuffer mode</p> 
	<p data-bbox="724 1482 1110 1512">b A short program will be fully loaded.</p> 

### 4.2.3 Program Management – Editing a program

Alterations to a textual program stored on the SIMATIC memory card can be performed using the Editor faceplate. Only programs which can be fully loaded can be edited, there is currently no editor support for ringbuffer mode.

Table 4-3 Program editing via WinCC

No.	Action
1.	 <p>The screenshot shows the 'SIMATIC WinCC Runtime Advanced' interface. On the left, there is a navigation pane with 'MBC' selected. The main area displays a list of programs on the 'SIMATIC Memory Card'. The programs listed are: 1. Bitman.mpf, 2. Demo.mpf, 3. Endless.mpf (highlighted), 4. Espresso.mpf, and 5. Italy2D.mpf. On the right side, there are buttons for 'Load', 'New', 'Copy', 'Transfer', and 'Delete'. The bottom navigation bar includes 'Program Manager', 'Editor', 'Program Control', and 'Portal'.</p>
2.	 <p>The screenshot shows the 'Program Editor' for 'Endless.mpf'. The program code is displayed in a list: 1. G17, 2. G54, 3. WHILE 1, 4. G91, 5. GO A90, 6. GO A-90, 7. G90, 8. GO X1200 (highlighted), 9. GO X1200, and 10. ENDWHILE. On the right side, there is an 'Edit' button (highlighted), an 'Insert commands' section with a 'New Line' button, and a 'Save program to card' button. The bottom navigation bar includes 'Program Manager', 'Editor', 'Program Control', and 'Portal'.</p>
3.	 <p>The screenshot shows the 'Program Editor' for 'Endless.mpf' with a virtual keyboard overlay. The program code is visible, and the 'GO X1200' line is selected. The virtual keyboard is displayed in the foreground, showing letters, numbers, and function keys. The bottom navigation bar includes 'Program Manager', 'Editor', 'Program Control', and 'Portal'.</p>

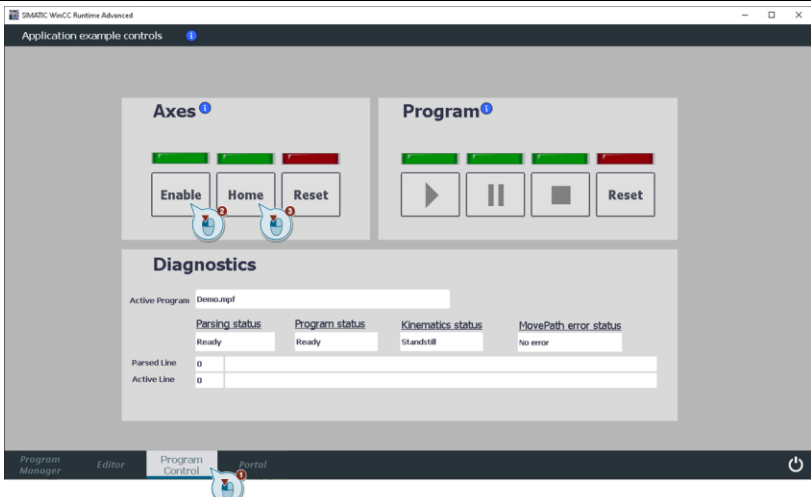
No.	Action
4.	

#### 4.2.4 Program execution

The program execution of the Kinematics Language bridge saw application is controlled from the HMI screen *Program Control*. A viable program needs to be selected.

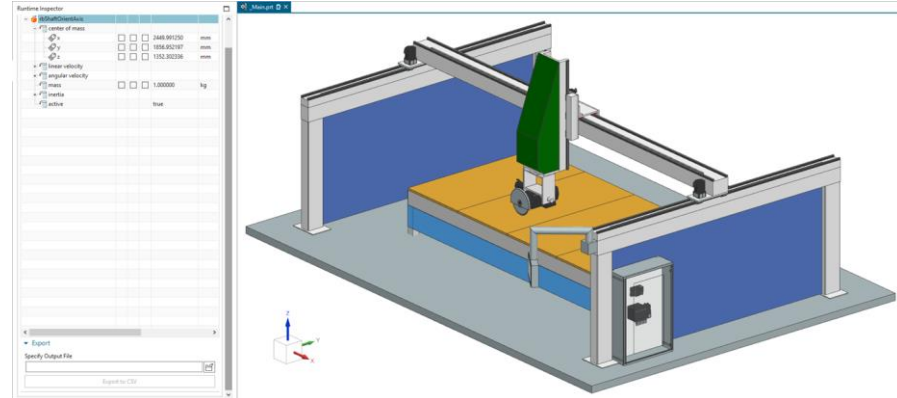
On initial start of the Software in the Loop twin, the safety acknowledgement and resetting of errors is required. The procedure is described in chapter [4.3.1](#).

Table 4-4 Execution of a textual program

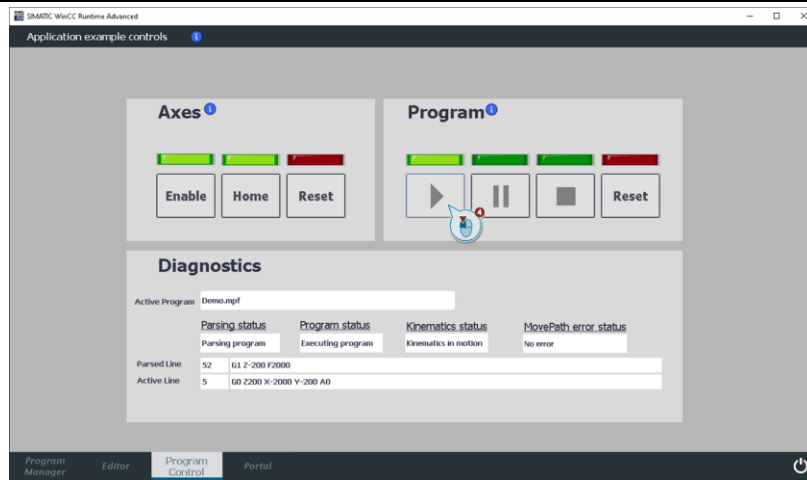
No.	Action
1.	

## 4 Operating the Digital Twin

2. This position corresponds with the zero position of the kinematic. The rigid body rbShaftOrientAxis equals the portal kinematic flange.

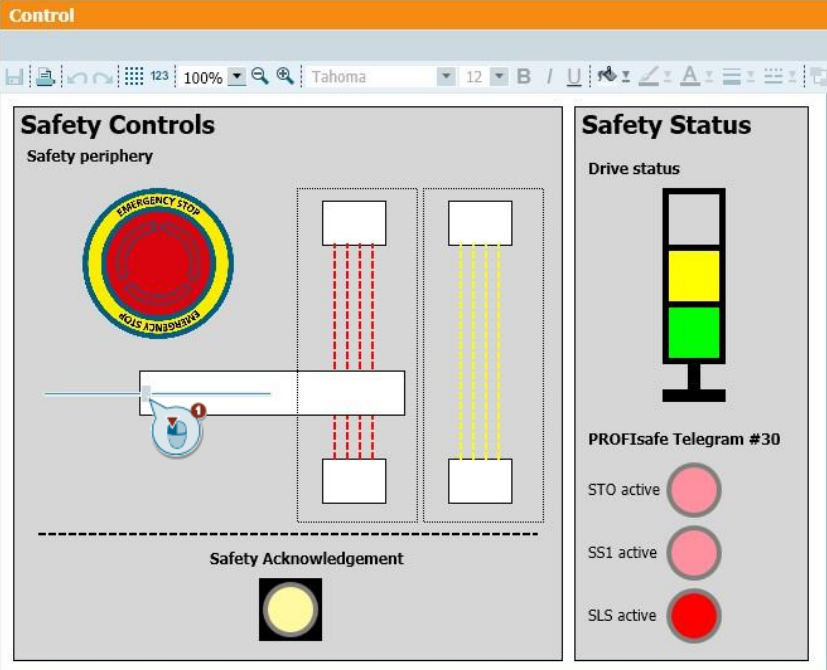
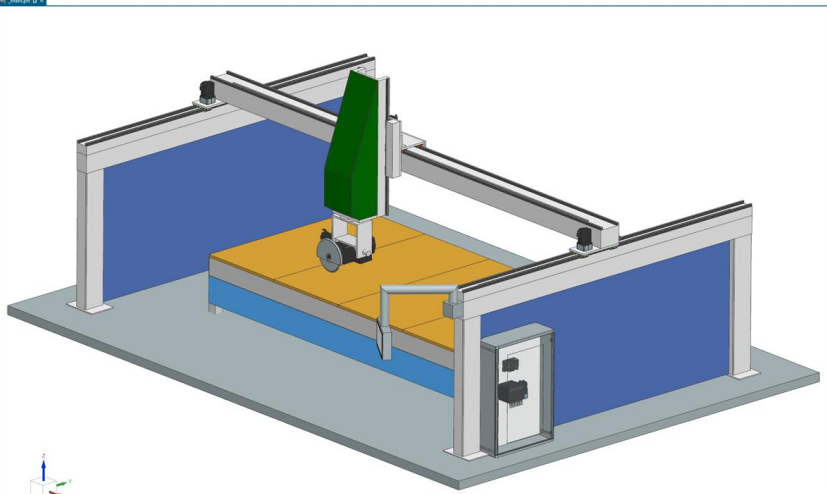


- 3.



### 4.2.5 Simulation of safety functionality

Table 4-5 Simulation of Safely Limited Speed and Safe Stop

No.	Action
1.	<p>Simulated penetration of <i>Light barrier #1</i> leads to Safely Limited Speed. The obstruction is created by moving the slider to the right.</p> 
2.	<p>Observe reduced axes and spindle speed in NX MCD model.</p> 



3. Simulated EStop button or penetration of Light barrier #1 AND Light barrier #2 lead to Safe Stop 1 (SS1).

4. Remove EStop and light barrier penetration. The further reset of the error state is described in chapter 4.3.1.

© Siemens AG 2021. All rights reserved

### 4.2.6 Manual control of kinematic

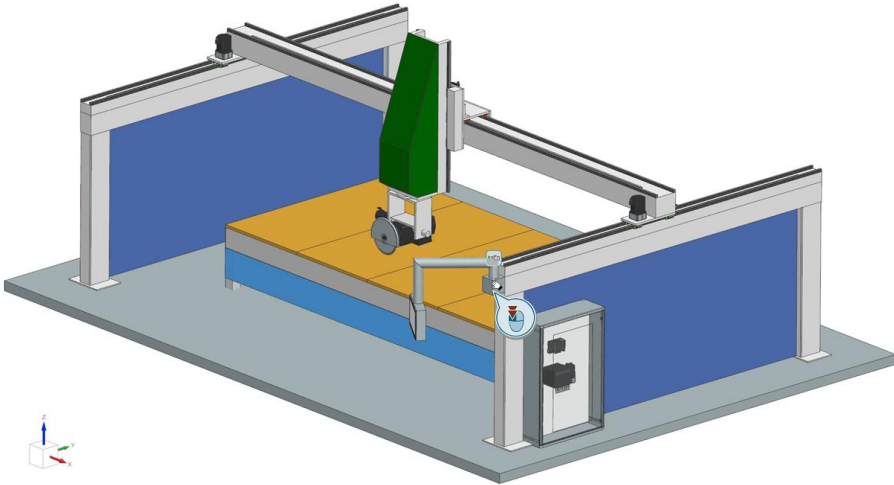
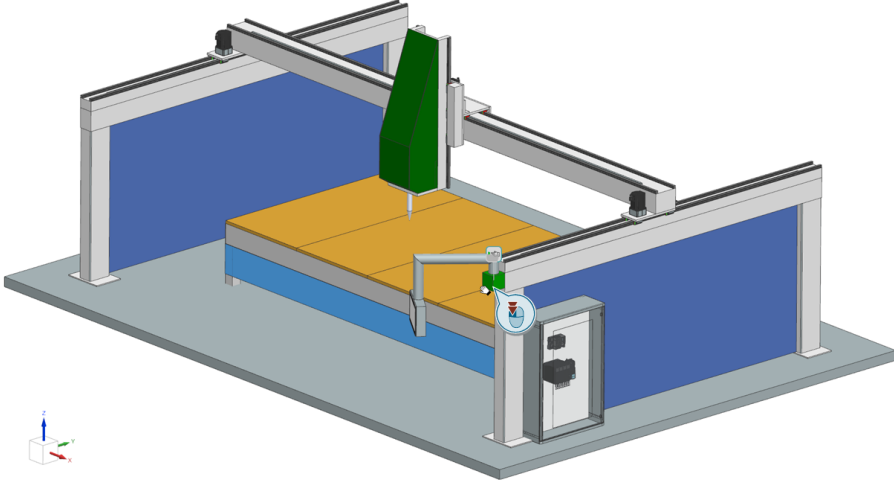
The application example allows the user to manually control the kinematics technology object. This is achieved by integrating the Kinematics Manual Control (LKinMCtrl) library and faceplate. The HMI screen is reached via the *Portal* tab in the application example HMI. If the axes are still enabled in the *Program Control* screen, the LKinMCtrl faceplate will operate in monitoring mode. When the axes have been disabled in *Program Control*, it is possible to gain control authority via the LKinMCtrl screen and use its full functionality. The detailed description is contained in [17](#).

**NOTE** The LKinMCtrl faceplate is limited to access to the TO Kinematic. Therefore, it does not provide a manual control possibility for the speed axis (spindle) or any auxiliary axis. If necessary, the required HMI screens can be added by the user. The LAxisBasics library can provide useful blocks and screens. It shares a SIOS entry with the LAxisCtrl library [15](#).

### 4.2.7 Toggle Sawblade ↔ Waterjet display

The saw blade in the generic bridge saw model does not allow for the sensible execution of very intricate and fine contours. To show the capabilities of the LKinLang application also for these purposes, the displayed tool in NX MCD can be changed from the saw blade to a generic “point-tool”. This addresses machining centers that rely on laser or water-jet cutting.

Table 4-6 Change the displayed tool in NX MCD

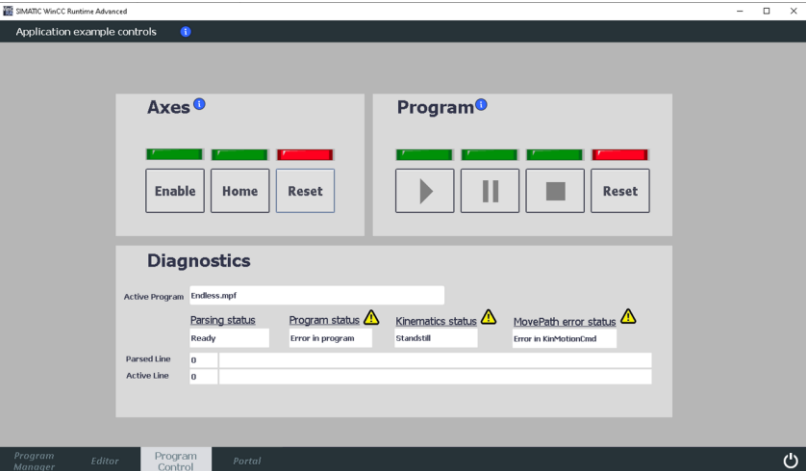
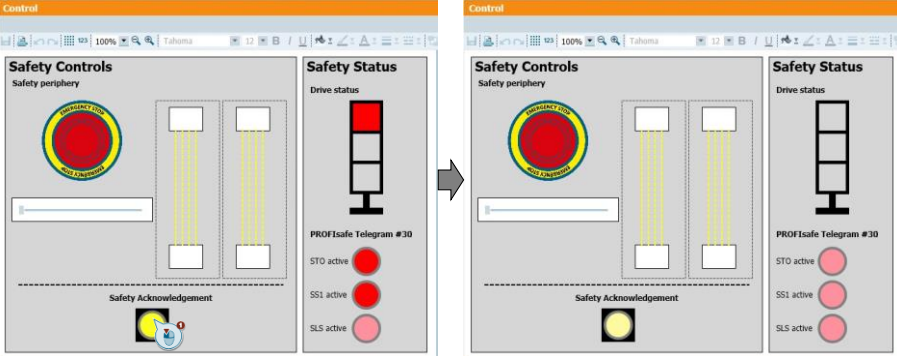

No.	Action
1.	 <p>The screenshot shows a 3D model of a bridge saw in a simulation environment. The tool is a large green saw blade. The machine has a blue frame and a yellow worktable. A control panel is visible on the right side.</p>
2.	 <p>The screenshot shows the same 3D model of the bridge saw. The tool has been changed from a saw blade to a small green point tool. The rest of the machine and environment are identical to the first screenshot.</p>

## 4.3 Error handling

### 4.3.1 Resetting of errors (startup & safety)

The acknowledgement and reset of errors are necessary when a safety integrated reaction has been performed. This is triggered by either the emergency stop or the violation of both light barrier simulations in the SIMIT *Control* diagram. Furthermore, the reset sequence needs to be executed once after simulation start.

Table 4-7 Safety acknowledgement and error reset

No.	Action
1.	 <p>The screenshot shows the SIMATIC WinCC Runtime Advanced interface. It features two main control panels: 'Axes' and 'Program'. The 'Axes' panel includes 'Enable', 'Home', and 'Reset' buttons. The 'Program' panel includes 'Start', 'Pause', 'Stop', and 'Reset' buttons. Below these is a 'Diagnostics' section with the following status indicators: Active Program: Endless.mpl; Parsing status: Ready; Program status: Error in program (with a yellow warning triangle); Kinematics status: Standstill (with a yellow warning triangle); MovePath_error status: Error in KinMotionCmd (with a yellow warning triangle). The bottom of the interface shows tabs for 'Program Manager', 'Editor', 'Program Control', and 'Portal'.</p>
2.	 <p>The diagram illustrates the safety status transition. On the left, the 'Safety Status' section shows 'STO active' (red), 'SSI active' (red), and 'SLS active' (pink). A 'Safety Acknowledgement' button is highlighted with a mouse cursor. An arrow points to the right, where the 'Safety Status' section shows 'STO active' (pink), 'SSI active' (pink), and 'SLS active' (pink), indicating that the safety system is now in a non-active state after acknowledgement.</p>
3.	 <p>This screenshot is similar to the first one, but the 'Reset' button in the 'Program' control panel is highlighted with a mouse cursor, indicating the next step in the error reset process.</p>

## 5 Appendix

### 5.1 Service and support

#### Industry Online Support

Do you have any questions or need assistance?

Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.

The Industry Online Support is the central address for information about our products, solutions and services.

Product information, manuals, downloads, FAQs, application examples and videos – all information is accessible with just a few mouse clicks:

[support.industry.siemens.com](https://support.industry.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:

[www.siemens.com/industry/supportrequest](https://www.siemens.com/industry/supportrequest)

#### SITRAIN – Training for Industry

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:

[www.siemens.com/sitrain](https://www.siemens.com/sitrain)

#### Service offer

Our range of services includes the following:

- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts

You can find detailed information on our range of services in the service catalog web page:

[support.industry.siemens.com/cs/sc](https://support.industry.siemens.com/cs/sc)

#### Industry Online Support app

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

[support.industry.siemens.com/cs/ww/en/sc/2067](https://support.industry.siemens.com/cs/ww/en/sc/2067)

## 5.2 Application support

Siemens AG  
 Digital Factory Division  
 Factory Automation  
 Production Machines  
 DF FA PMA APC  
 Fraunauracher Str. 80  
 91056 Erlangen, Germany  
 mailto: [tech.team.motioncontrol@siemens.com](mailto:tech.team.motioncontrol@siemens.com)

## 5.3 Links and literature

Table 5-1

No.	Topic
\1\	Siemens Industry Online Support <a href="https://support.industry.siemens.com">https://support.industry.siemens.com</a>
\2\	Link to this entry page of this application example <a href="https://support.industry.siemens.com/cs/ww/en/view/109804122">https://support.industry.siemens.com/cs/ww/en/view/109804122</a>
\3\	SIMATIC S7-1500T Kinematics Language <a href="https://support.industry.siemens.com/cs/document/109767009">https://support.industry.siemens.com/cs/document/109767009</a>
\4\	SIMATIC S7-1500T Kinematics Control <a href="https://support.industry.siemens.com/cs/document/109755891">https://support.industry.siemens.com/cs/document/109755891</a>
\5\	SIMATIC S7-1500/S7-1500T Standard application axis control <a href="https://support.industry.siemens.com/cs/document/109749348">https://support.industry.siemens.com/cs/document/109749348</a>
\6\	Simulation Model Generator <a href="https://support.industry.siemens.com/cs/document/109780391">https://support.industry.siemens.com/cs/document/109780391</a>
\7\	SIMATIC S7-1500T Kinematics Manual Control <a href="https://support.industry.siemens.com/cs/document/109755892">https://support.industry.siemens.com/cs/document/109755892</a>
\8\	SIMIT / MCD / AMESIM – Virtual commissioning in machine building <a href="https://support.industry.siemens.com/cs/document/109777165">https://support.industry.siemens.com/cs/document/109777165</a>
\9\	SIMATIC – Failsafe library LDrvSafe to control the Safety Integrated functions of the SINAMICS drive family <a href="https://support.industry.siemens.com/cs/document/109485794">https://support.industry.siemens.com/cs/document/109485794</a>
\10\	S7-PLCSIM Advanced V4.0 - Correction of the V3.0 API version <a href="https://support.industry.siemens.com/cs/document/109802065">https://support.industry.siemens.com/cs/document/109802065</a>

## 5.4 Change documentation

Table 5-2

Version	Date	Modifications
V1.0	11/2021	First version

